Understanding Deep Learning

Dissertation

zur Erlangung des akademischen Grades
doctor rerum naturalium (Dr. rer. nat.)

vorgelegt dem Rat der Fakultät für Mathematik und Informatik
der Friedrich-Schiller-Universität Jena

von M.Sc. Christian Reimers
geboren am 26. Oktober 1991 in Hameln.

Gutachter:

1. Prof. Dr.-Ing. Joachim Denzler,
   Friedrich-Schiller-Universität Jena, Jena, Germany
2. Prof. Dr. Jürgen Gall,
   Universität Bonn, Bonn, Germany

Tag der Einreichung: 11. Juli 2022
Tag der öffentlichen Verteidigung: 11. Januar 2023

Acknowledgments

I would like to express my deepest gratitude to Joachim Denzler for his supervision, suggestions, feedback and inputs. Further, this work would not have been possible without the support and feedback of Jakob Runge and Paul Bodesheim. Also I would like to express my deepest appreciation to the reviewers and members of the review board.
I would also like to thank my colleagues in the Computer Vision Group at the Friedrich Schiller University Jena and the Climate Informatics Group at the Institute of Data Science of the German Aerospace Center for many interesting and fruitful discussions, but also for their great support.
Last but not least, I would like to thank Jessica and my family for their emotional support.

Short Summary

Deep neural networks have reached an impressive performance in many tasks in computer vision and its applications. Further, many of the methods developed in computer vision are transferrable to other kinds of data, such as time series or text. However, one prominent feature of deep learning is automatic feature selection. If we use a deep neural network to make predictions, it is nontrivial to understand which features the deep learning approach extracted from the data or which features the deep neural network used to reach its prediction. Nevertheless, this information is essential in some applications, for example, safety or security-critical applications, where we expect the user to trust the decision of a neural network with their life or well-being. As a specific example, we focus on dermoscopic image analysis in this work. Another area in which it is crucial to understand the features a neural network uses in tasks where we want to use it to further our understanding of a system. We want to use a neural network to understand connections between variables and are less interested in the correct predictions in these tasks. Examples of a field with such tasks are climate and earth-system science.

Research into understanding deep learning is challenging due to the evaluation. Since it is unknown which features deep neural networks use, it is hard to empirically evaluate whether a result for which feature is used by a deep neural network is correct. One effect that highlights our lack of understanding of deep neural networks is adversarial examples. Given an input image classified correctly by a classifier, an adversary can provoke a misclassification of the automatic classifier by adding a carefully calculated but imperceptibly small noise to the image. The resulting image is called an adversarial example. Since the imperceptibly small perturbation does not change any of the features a human would deem relevant this demonstrates, that deep neural networks do not rely only on the same features that humans would use. In this work, we start by furthering the understanding of adversarial examples. The main focus in that part of this work is to find a suitable definition of adversarial examples that allows us to differentiate between the intriguing effect of adversarial examples and the not at all intriguing observation, that we can alter the decision of an automatic classifier by changing the content of the input image. We use this definition to further our understanding of why adversarial examples exist. In particular, we demonstrate that the modality of the data distribution impacts the vulnerability of classifiers trained on the data.

The state-of-the-art for understanding which features a deep neural network

uses to reach its prediction is sailiency maps. A sailiency map is a mapping that assigns an importance value to each pixel of the input image. Consequently, it can highlight important areas of the input image. The main ways to create these sailiency maps are the gradient of the function represented by the deep neural network, obfuscation of regions of the input image and Taylor-approximation of the mapping from perturbation to change in output. For the gradient method, the gradient of the function that maps the input images onto the prediction of the deep neural network given the input pixel is calculated. This gradient is used either directly as sailiency or slightly modified. For the obfuscation method, we obfuscate parts of the input image and recalculate the classifier's prediction. The sailiency is the difference between the original prediction and the prediction containing the obfuscation. The Taylor approximation method is an intermediate idea between the two. The idea is to approximate the function that maps the perturbation onto the change in the classifier's prediction using a first-order Taylor approximation. However, all methods built on sailiency maps share shortcomings that open a gap between the current state-of-the-art and the requirements for understanding deep neural networks. First, non of the three methods mentioned above are intrinsically linked to an input influencing an output. Second, since semantic parts of the image are in different positions in different images, it is challenging to generalize observations made on single samples to the level of the classifier as a whole. Third, and most importantly, since sailiency maps highlight areas of the input, they can only be used for features represented by areas of the input. For example, features that are parts of objects, such as the head of a bird, are represented by areas of the input, while properties of the whole object, for example, the color of a bird, are not.

This work describes a method that does not suffer from these shortcomings. To this end, we employ the framework of causal modeling. The framework of causal modeling arranges the variables and processes of a system into a directed acyclic graph where the variables form the nodes, and the processes form the edges. Such a representation of a system is called a structural causal model. In this model, a cause influences an effect if and only if a directed path in the graph connects the cause to the effect. Using this framework, we represent supervised learning as a structural causal model. To check whether a feature influences the prediction of a classifier, we block all paths, except the one representing the inference function of the classifier, by conditioning on variables along the respective paths. We test, if, even after conditioning on these variables, the feature and the prediction of the classifier are dependent. If the result is affirmative, the feature influences the classifier's prediction through the inference function. In other words, the classifier extracts and uses the feature. We demonstrate that this method can understand whether a supervised learning classifier uses a feature. To this end, we, on the one hand, test shallow classifiers for which we can validate our findings, and, on the other hand, deep learning classifiers for which we can only conclude that our method returns plausible and meaningful results.

We further demonstrate two applications of our method. First, we show that our method can further the understanding of automatic skin lesion classifiers. Skin

cancer is a very deadly diseases, and early detection is vital in treating it. Since early detection requires regular checks by medical professionals, they are labor extensive. Automatic classification of skin lesions can support practitioners and make early detection feasible. Visual analysis is a common first step in skin lesion classification, and dermatologists have developed features to determine whether a skin lesion is malignant. These features are named in the dermoscopic ABCD rule, an algorithm developed to identify melanomas. We investigate multiple state-of-the-art classifiers and determine whether they use the features named in the dermoscopic ABCD rule. Further, we investigate whether the same classifiers rely on bias variables, namely the patient's age, sex, and skin color and the existence of colorful patches in the input image.

We find that some of the features in the ABCD rule are used by the classifiers to identify melanoma but not to identify seborrheic keratosis. In contrast, all classifiers highly rely on the bias variables, particularly the age of the patient and the existence of colorful patches in the input image.

The second application is adversarial debiasing. In adversarial debiasing, we want to stop a neural network from using a known bias variable. To this end, the idea is to use a second loss next to the classification loss. This debiasing loss punishes the deep neural network for using the bias feature. Since an obvious first step in this process is to determine whether the deep neural network uses a feature, our work applies. The state-of-the-art in adversarial debiasing is to enforce independence between the bias variable and the classifier's prediction. Building on the work on determining whether a classifier uses a feature, we propose to use the conditional dependence conditioned on the ground truth instead. We prove mathematically that under reasonable assumptions on the bias creation, an optimal classifier fulfills this conditional dependence, while it does not fulfill the unconditional independence used by the current state-of-the-art in adversarial debiasing. Further, we demonstrate in a toy example and an example on real-world images that our approach outperforms the state-of-the-art in adversarial debiasing.

In conclusion, we make deep neural networks more transparent and, consequently, more robust. To this end, we demonstrate the current challenges in understanding deep neural networks using the phenomenon of adversarial examples. We employ causality and the framework of structural causal models to determine whether a deep neural network uses a feature. We demonstrate that this method can determine whether an automatic classifier uses a feature. We demonstrate that the resulting method is applicable and valuable by applying it to the problem of skin lesion classification understanding. We develop a new, improved method for adversarial debiasing based on our new method to determine whether the classifier uses a feature and demonstrate that this method outperforms the state-of-the-art from the literature.

Kurzzusammenfassung

Tiefe neuronale Netze haben in vielen Aufgaben in Forschungsbereichen des Computersehens beeindruckende Ergebnisse erzielt. Darüber hinaus lassen sich viele der im Computersehen entwickelten Methoden auch auf andere Arten von Daten wie Zeitreihen oder Text übertragen. Ein herausragendes Merkmal tiefer Learnverfahren ist die automatische Auswahl von Merkmalen. Wenn wir ein tiefes neuronales Netz verwenden, um Vorhersagen zu treffen, ist es nicht trivial zu verstehen, welche Merkmale das tiefe Lernverfahren aus den Daten extrahiert hat und welche Merkmale das tiefe neuronale Netz verwendet hat, um seine Vorhersage zu treffen. Diese Informationen sind jedoch bei einigen Anwendungen von entscheidender Bedeutung, z.B. bei sicherheitskritischen Anwendungen, bei denen wir erwarten, dass der Benutzer der Entscheidung eines neuronalen Netzes sein Leben oder sein Wohlergehen anvertraut. Als konkretes Beispiel konzentrieren wir uns in dieser Arbeit auf die dermatoskopische Bildanalyse. Ein weiterer Bereich, in dem es von entscheidender Bedeutung ist, die Merkmale zu verstehen, die ein neuronales Netz verwendet sind Situationen in denen wir ein neuronales Netz verwenden um unser Verständnis eines Systems zu verbessern. In denen wir ein neuronales Netz verwenden, um die Verbindungen zwischen Variablen zu verstehen, und weniger an den korrekten Vorhersagen interessiert sind. Beispiele für solch ein Gebiet sind Klima- und Erdsystemwissenschaften.

Die Forschung zum Verständnis von tiefen Lernverfahren ist auch aufgrund der Evaluation eine Herausforderung. Da nicht bekannt ist, welche Merkmale tiefe neuronale Netze verwenden, ist es schwierig, empirisch zu bewerten, ob das Ergebnis, welches Merkmal von einem tiefen neuronalen Netz verwendet wird, correct ist. Ein Effekt, der unser mangelndes Verständnis von tiefen neuronalen Netzen hervorhebt, sind “adversarial examples”. Hierbei kann ein Angreifer eine Fehlklassifizierung eines zuvor korrect klassifizierten Eingabebildes provozieren. Dazu addiert er ein sorgfältig berechnetes, aber kaum wahrnehmbares Rauschens zum Bild. Das resultierende Bild wird als “adversarial example” bezeichnet. Da die kaum wahrnehmbare Störung kein einziges der Merkmale verändert, die ein Mensch für relevant halten würde, zeigt dies, dass tiefe neuronale Netze sich nicht ausschließlich auf die gleichen Merkmale stützen, die Menschen verwenden. In dieser Arbeit beginnen wir mit der Vertiefung des Verständnisses von “adversarial examples”. Das Hauptaugenmerk in diesem Teil der Arbeit liegt auf der Suche nach einer geeigneten Definition von “adversarial examples” zu finden, die es uns

ermöglicht, zwischen dem faszinierenden Phänomen der “adversarial example” und nicht verblüffenden Beobachtung, dass wir die Entscheidung eines automatischen Klassifizierers ändern können, indem wir den Inhalt des Eingabebildes ausreichend ändern. Wir verwenden diese Definition, um besser zu verstehen, warum “adversarial examples” existieren. Insbesondere zeigen wir, dass die Modalität der Datenverteilung die Robustheit der auf den Daten trainierten Klassifikatoren auswirkt.

Der Stand der Technik, um zu verstehen, welche Merkmale ein tiefes neuronales Netz verwendet, um seine Vorhersage zu erreichen, sind “Sailiency Maps”. Eine “Sailiency Map” ist eine Abbildung, die jedem Pixel des Eingabebildes einen Wichtigkeitswert zuweist. Folglich kann sie wichtige Bereiche des Eingabebildes hervorheben. Die wichtigsten Methoden zur Erstellung dieser “Sailiency Maps” sind der Gradient der Funktion, die durch das tiefe neuronale Netz dargestellt wird, das Austauschen von Regionen des Eingabebildes und die Taylor-Approximation der Abbildung von der Störung auf die Veränderung der Ausgabe des neuronalen Netzes. Bei der Gradientenmethode wird der Gradient der Funktion, die die Eingabebilder auf die Vorhersage des tiefen neuronalen Netzes abbildet, berechnet. Dieser Gradient wird entweder direkt als Wichtigkeit verwendet oder leicht modifiziert. Bei der Austauschmethode werden Teile des Eingangsbildes ausgetauscht und die Vorhersage des Klassifikators neu berechnet. Die Wichtigkeit ergibt sich aus der Differenz zwischen der ursprünglichen Vorhersage und der Vorhersage für das geänderte Bild. Die Taylor-Approximationsmethode ist eine Zwischenlösung zwischen den beiden Verfahren. Die Idee besteht darin, die Funktion, die die Störung auf die Änderung der Vorhersage des Klassifikators abbildet, durch eine Taylor-Approximation erster Ordnung zu approximieren. Alle Methoden, die auf “Sailiency Maps” aufbauen, weisen jedoch Schwächen auf, die eine Lücke zwischen dem aktuellen Stand der Technik und den Anforderungen an das Verständnis tiefer neuronaler Netze offenbaren. Erstens ist keine der drei oben genannten Methoden natürlich mit der Wichtigkeit einer Eingabe verbunden. Zweitens, da semantische Teile des Bildes in verschiedenen Bildern an unterschiedlichen Positionen befinden, ist es schwierig, Beobachtungen, die an einzelnen Proben gemacht wurden, auf die Ebene des Klassifikators als Ganzes zu verallgemeinern. Aber vor allem, drittens, da “Sailiency Maps” nur Bereiche des Inputs hervorheben, können sie nur für Merkmale verwendet werden, die durch eindeutige Bereiche des Inputs repräsentiert werden. Beispiele für Merkmale die durch Bereiche des Bildes eindeutig repräsentiert werden sind Teile von Objekten sind, wie z. B. der Kopf eines Vogels, während Eigenschaften des gesamten Objekts, z. B. die Farbe eines Vogels, nicht durch Regionen eindeutig repräsentiert werden.

In dieser Arbeit beschreiben wir eine Methode, die diese Schwächen nicht teilt. Dazu verwenden wir Methoden der kausalen Modellierung. Kausale Modellierung ordnet die Variablen und Prozesse eines Systems in einem gerichteten azyklischen Graphen an, wobei die Variablen die Knoten und die Prozesse die Kanten bilden. Eine solche Darstellung eines Systems wird als strukturelles Kausalmodell bezeichnet. In diesem Modell beeinflusst eine Ursache eine Wirkung genau dann, wenn ein

gerichteter Pfad im Graphen die Ursache mit der Wirkung verbindet. Wir stellen das überwachte Lernen als strukturelles Kausalmodell dar. Um zu überprüfen, ob ein Merkmal die Vorhersage eines Klassifikators beeinflusst, blockieren wir alle Pfade mit Ausnahme des Pfades, der die Inferenzfunktion des Klassifikators darstellt, indem wir die Variablen entlang der jeweiligen Pfade bedingen. Wir testen, ob auch nach der Bedingung auf diese Variablen das Merkmal und die Vorhersage des Klassifikators voneinander abhängig sind. Wenn das Ergebnis positiv ist, beeinflusst das Merkmal die Vorhersage des Klassifikators durch die Inferenzfunktion. Mit anderen Worten: Der Klassifikator extrahiert und verwendet das Merkmal. Wir zeigen, dass diese Methode bestimmen kann ob ein Klassifikator ein Merkmal verwendet. Zu diesem Zweck testen wir zum einen einfache Klassifikatoren, für die wir unsere Ergebnisse validieren können, und andererseits tiefe neuronale Netze, für die wir nur feststellen können, dass unsere Methode plausible und aussagekräftige Ergebnisse liefert.

Wir demonstrieren außerdem zwei Anwendungen unserer Methode. Erstens zeigen wir, dass unsere Methode das Verständnis für die automatische Klassifizierung von Hautläsionen verbessern kann. Hautkrebs ist eine sehr tödliche Krankheit, deren frühzeitige Erkennung für die Behandlung entscheidend ist. Da die Früherkennung regelmäßige Kontrollen durch medizinisches Fachpersonal erfordert, ist sie sehr arbeitsintensiv. Die automatische Klassifizierung von Hautläsionen kann Ärzte unterstützen und eine Früherkennung ermöglichen. Die visuelle Analyse ist ein üblicher erster Schritt bei der Klassifizierung von Hautläsionen, und Dermatologen haben Merkmale entwickelt, um festzustellen, ob eine Hautläsion bösartig ist. Diese Merkmale werden in der dermatoskopischen ABCD-Regel genannt, einem Algorithmus, der zur Erkennung von Melanomen entwickelt wurde. Wir untersuchen mehrere moderne Klassifikatoren und ermitteln, ob sie die in der dermatoskopischen ABCD-Regel genannten Merkmale verwenden. Darüber hinaus untersuchen wir, ob dieselben Klassifikatoren auf Stögrößen wie Alter, Geschlecht und Hautfarbe des Patienten oder das Vorhandensein farbiger Pflaster im Eingabebild.

Wir stellen fest, dass einige der Merkmale in der ABCD-Regel von den Klassifikatoren verwendet werden, um Melanome zu identifizieren, aber nicht, um seborrhoische Keratose zu identifizieren. Im Gegensatz dazu verlassen sich alle Klassifikatoren in hohem Maße auf die Biasvariablen, insbesondere auf das Alter der Patienten und das Vorhandensein farbiger Pflaster im Eingangsbild.

Die zweite Anwendung ist “adversarial debiasing”. Beim “adversarial debiasing” soll verhindert werden, dass ein neuronales Netz eine bekannte Biasvariable verwendet. Zu diesem Zweck wird eine zweite Zielfunktion neben dem Klassifikationsverlust verwendet. Dieser Debiasing-Verlust bestraft das tiefe neuronale Netz dafür, dass es die Biasvariable verwendet. Da ein offensichtlicher erster Schritt in diesem Prozess darin besteht, festzustellen, ob das tiefe neuronale Netz ein Merkmal verwendet, findet unsere Arbeit Anwendung. Der Stand der Technik bei “adversarial debiasing” besteht darin, die Unabhängigkeit zwischen der Biasvariable und der Vorhersage des Klassifikators zu erzwingen. Aufbauend auf den Arbeiten zur Bestimmung, ob ein Klassifikator ein Merkmal verwendet, schlagen wir vor, stattdessen

die bedingte Abhängigkeit, bedingt auf die richtige Klasse zu verwenden. Wir beweisen mathematisch, dass ein optimaler Klassifikator unter vernünftigen Annahmen bezüglich der Bias-Erzeugung diese bedingte Abhängigkeit erfüllt, während er die bedingungslose Unabhängigkeit nicht erfüllt, die von anderen Arbeiten im Bereich des “adversarial debiasing” verwendet wird. Darüber hinaus zeigen wir anhand eines künstlichem und eines Beispiels auf Fotos, dass unser Ansatz den Stand der Technik im “adversarial debiasing” übertrifft.

Zusammenfassend lässt sich sagen, dass wir tiefe neuronale Netze transparenter und folglich robuster machen. Zu diesem Zweck zeigen wir die aktuellen Herausforderungen beim Verständnis tiefer neuronaler Netze anhand des Phänomens der “adversarial examples”. Wir verwenden Kausalität und struktureller Kausalmodelle, um festzustellen, ob ein tiefes neuronales Netz ein Merkmal verwendet. Wir zeigen, dass diese Methode feststellen kann, ob ein automatischer Klassifikator ein Merkmal verwendet. Wir zeigen, dass die daraus resultierende Methode anwendbar und wertvoll ist, indem wir sie auf das Problem des Verständnisses der Klassifizierung von Hautläsionen anwenden. Wir entwickeln ein neues, verbessertes Verfahren für “adversarial debiasing”, das auf unserer neuen Methode basiert, und zeigen, dass diese Methode andere Methoden aus der Literatur übertrifft.

Contents

1 | Introduction

In recent years, deep neural networks have reached impressive performance and have superseded classical machine learning methods for many tasks in computer vision. The adaption of deep learning brought many advantages but also rised new challenges in these tasks. To illustrate these advantages and new challenges, we use automatic skin lesion classification as an example throughout this introduction. The task in skin lesion classification is to predict whether a skin lesion is malignant from an image of the skin lesion. For a further introduction into this relevant, challenging task, we refer the reader to Section 1.2.

The main structure of classical machine learning algorithms for classifying skin lesions stayed the same for a long time. It is displayed and summarized in Celebi et al. (2007). The algorithms include four steps. In the first step, the algorithm divides the image into the lesion and the image's background. The second step is to extract the skin lesion's handcrafted, medically relevant features. Candidates for these features are, for example given by the ABCD rule (Nachbar et al., 1994; Stolz and Kunz, 2021) or by the seven-point checklist (Bahmer et al., 1990). The third step is to select a subset of these variables with a high predictive power for the classification task. Finally, a classifier is trained on these features in the fourth step. This methodology produced classifiers that reached classification performance comparable to practitioners in practice as investigated, for example, by Hoffmann et al. (2003). Not only was this multistep procedure state-of-the-art in skin lesion classification, but it is representative of shallow machine learning methods in many tasks. A significant advantage of this approach is its interpretability. Since the features are handcrafted and relevant from the perspective of the domain, a simple, interpretable classifier achieves excellent results.

However, deep learning approaches have proven to be even more effective in recent years. Deep learning classifiers condense the multiple steps of shallow learning classifiers into one end-to-end optimization. This condensation increases the method's effectiveness but these performance improvements come at a cost. First, the method requires more images and computing capabilities, and, since the features are selected automatically by the deep learning algorithm, it is non-trivial to understand which features it selects.

In the example of skin lesion classification, large datasets are provided, for example, by Tschandl et al. (2018), Combalia et al. (2019) and Codella et al. (2018). Newer deep learning classifiers condense the four abovementioned steps into a single

training step. A large ensemble of deep neural networks is used as the classifier. These networks are pre-trained on large image datasets, for example, ImageNet (Russakovsky et al., 2015) and employ heavy test time augmentation (Perez et al., 2018). An example for such an ensemble classifier is described in Gessert et al. (2020). The performance of these deep learning ensemble classifiers is impressive. A study conducted by Tschandl et al. (2019) compared the performance of the best algorithms from the ISCI 2018 challenge (Codella et al., 2019) with experienced practitioners. They presented human readers with batches of thirty images that had to be classified as one of seven predefined classes. They found that deep learning algorithms outperformed even dermatologists with more than ten years of experience by a margin of 22 percentage points. However, the authors can no longer tell which features the classifier uses, which motivates further research (Tschandl et al., 2020).

Nevertheless, understanding which feature is used by an automatic classifier is important in many tasks. This importance is highlighted by two example groups of tasks where it is crucial to understand which feature is used by the classifier. The first group is safety and security-critical tasks. This group includes not only medical tasks such as the one explained in the example above but also tasks such as the prediction of recidivism as employed in the USA (Barry-Jester et al., 2015) or driver assistance systems that, for example, predict the aquaplaning risk from a camera image as proposed, for example, by Schneider et al. (2018). In all of these applications, people trust the predictions of deep learning algorithms with their live, well-being or freedom. Therefore, it is important to understand which feature is used by the automatic classifiers that make these decisions. For example, Barry-Jester et al. (2015) and Larson et al. (2016) express concerns that Northpoint's COMPAS[1] system uses the membership of an ethnic group.

The second group of classifiers for which it is important to understanding which features are used by a deep neural network are classifiers in scientific areas where we want to use their ability to detect informative features as well as find and understand links in the data. An example for such an area is climate and earth-system science. In climate science, classifiers often focus on understanding the system rather than predicting it accurately. In many ways, deep learning is a good fit for climate science. The amount of available data is immense, the data is highly autocorrelated, and we cannot conduct large-scale controlled experiments. However, the focus is often not to predict variables as precisely as possible but to understand the data. As described above, the high complexity of deep neural networks and the fact that they are best trained end-to-end makes it challenging to understand the predictions of deep neural networks.However,

1. most tasks in climate science are not pure prediction or classification tasks. Instead, they focus on understanding which features in the data are important and how variables are connected.

2. Physics aims to understand micro-processes and infer the behavior of macro

[1] www.northpointeinc.com

processes by the laws of statistics. Hence, micro-processes are precisely defined, while macro processes are often inherently stochastic and fuzzily defined. Therefore, the vast knowledge on these macro processes is hard to present by labeled data points. This challenge necessitates a method to evaluate whether these macro processes are respected and correctly modeled in the deep learning models, including their connections.

3. Climate Science is often concerned with predicting future scenarios that have never existed. While it is common to evaluate deep neural networks on hold-out test data, measurements for these scenarios do not exist. Hence, we need other ways to evaluate models in this situation. Understanding deep neural networks and identifying which features they use in their decision-making process can be an alternative way to evaluate deep neural networks.

4. The different climate and earth-system science variables are often heavily interconnected and correlated. These connections make it difficult to develop systems that ignore certain features of a situation. Due to the dependence between variables, it is not enough to omit a variable from the input, but we must actively correct other variables. The first step towards enforcing that a classifier ignores a variable is to measure whether the classifier uses it.

The main goal of this work is to further our understanding of which features are used by a deep neural network. More specifically, the focus of this work is to understand whether the deep neural network uses a specific, previously known feature. This task is, however, challenging. One observation that demonstrates the missing understanding in this area is adversarial examples. An adversarial example is an input image to an automatic classifier that an adversary manipulates to fool the automatic classifier by adding a perturbation that is imperceptible to a human observer. An example of adversarial examples can be found in Figure 2.2. Since the perturbation is imperceptible to a human observer, it can not change any input image feature that a human would consider relevant. Consequently, the fact that this manipulated input is classified wrong by the automatic classifier, even though all features relevant to a human are identical to the original input, proves that the classifier considers different features than a human. For this reason, we start our investigation with a study on adversarial examples. We investigate ways to measure whether an example is an adversarial example objectively. We present theoretical advances and empirical experiments on the reasons why adversarial examples exist in Section 2.5 and Section 3.

Adversarial examples demonstrate that it is non-trivial to find which feature is used by a deep neural network. To tackle this challenge, we developed a method to determine whether an automatic classifier uses a feature. While the method applies to any supervised learning algorithm, we focus primarily on deep neural networks in this work.

One of the main challenges when developing an algorithm to determine which feature a deep neural network uses is that there is no generally agreed-on method against which to evaluate the algorithm. Hence, it is crucial to base the algorithm on

a solid theoretical foundation. To this end, we base our method on the framework of causal modeling introduced by Pearl (2009) and Peters et al. (2017). This framework is specifically designed to answer questions about the influence of random variables on each other and is, hence, a good fit for the question we want to tackle. We provide a short introduction into the field of causal modeling as well as references to further reading and discussion of the framework in Section 2.1.

One of the cornerstones of the causal modeling framework of Pearl (2009) and Peters et al. (2017) is Reichenbach's common cause principle (Reichenbach, 1991). This principle states that if two variables are correlated or more generally statistically dependent, either one of them is causing the other or there exists a third variable, which we call a confounder, that is causing both of the original variables. This principle allows us to connect whether a variable is causing another variable, or, in other words, whether a variable is used to determine another variable to the result of a mathematical independence test. By considering not only the resulting classifier but the whole pipeline of supervised learning, this framework allows us to construct a causal graph that represents the processes and variables involved in supervised learning. These processes and variables are the distribution of the examples of a specific label P_L, the training set TS, the weights of the supervised learning algorithm W, the feature of interest X, the set of features $\bar{X}$ that are orthogonal to the feature of interest as well as the prediction P of the supervised learning algorithm, the sampling processes S_T and S_F to sample the training set or to sample single examples for inference, respectively, as well as the training process T and the inference process F. In Section 4.1.2 we explain how these variables and processes form the graphical model depicted in Figure 4.3. Using that graphical model, we can reduce the question of whether a deep neural network uses a feature to a conditional independence test, namely the test

$$X \perp\!\!\!\perp P \,|\, L$$

whether the feature X we investigate and the prediction P of the supervised learning algorithm are dependent given the ground-truth label L of the input sample. Hence, we reduced the challenging question of whether a deep neural network uses a feature down to a simple mathematical dependence test. This solution has excellent properties. First, it has a solid theoretical foundation in causal modeling and causal inference. Due to its simplicity, it applies to black-box deep neural network classifiers. Neither retraining nor intermediate results of the classifier are needed. These properties allow the use of our method even for users outside of the domain of deep learning experts, for example, domain experts from the domain of the prediction or classification task. Further, our method is not specific to deep neural networks but can instead be used for any supervised learning algorithm, for example, the random forest (Ho, 1995) or the k nearest-neighbor classifiers (Altman, 1992). More importantly, since it is not specific to any supervised learning classifier, it will most likely be valid for classifiers developed in the future.

A simple proof of concept on synthetic data is published in Reimers et al. (2019).

Subsequent research demonstrates that our method applies to big data sets composed of real-life images. These datasets include MSCOCO (Lin et al., 2014), a data set of photographs of everyday situations and crowded scenes containing multiple objects in every image, CUB200 (Welinder et al., 2010), a fine-grained bird recognition data set and HAM10000 (Tschandl et al., 2018), a medical datasets of skin lesions images. Experiments on these datasets, together with a thorough discussion of the underlying theory, are published in Reimers et al. (2020). We describe the theoretical considerations in Section 4 and the experiments in Section 4.3.1. In two studies, we further demonstrated the usefulness of this new method to determine whether a deep neural network uses a feature. We present the first of the two applications in Section 4.2.2. This application investigates which features a state-of-the-art automatic skin lesion classifier uses to determine whether a skin lesion is, for example, melanoma or seborrheic keratosis. To this end, we trained two groups of deep neural networks recognizing different classes of skin lesions following the training described in Perez et al. (2018) and Gessert et al. (2020). For all classifiers in both groups, we evaluate whether they use any out of four groups of features. The first group of features is features that have little to no information on the skin lesion itself. The fact that our method does not indicate that any of the groups of classifiers use any of the features indicates that our method is suitable for this complex real-world medical dataset. Furthermore, it indicates that in the cases where our method indicates that a feature is used, the classifier actually uses this feature.

The second group of features contains medically relevant features. When dermatologists decide whether a skin lesion is a melanoma, they rely on four features named in the dermoscopic ABCD rule introduced by Nachbar et al. (1994). To increase the trust in deep neural network classifiers for automatic skin lesion classifiers, we test whether they also use these features. The four features in the ABCD-rule are the number of orthogonal symmetry axes in the contour of the image, the sharpness and clearness of the border of the skin lesion, the variation in colors among different regions of the skin lesion and the presence of predefined dermoscopic features in the skin lesion, namely milia like cysts, negative networks, pigment networks, streaks, and globules. We find that the feature that concerns the symmetry and the feature that concerns the border of the skin lesions is used by classifiers trained to recognize melanoma but not by classifiers that recognize seborrheic keratosis. This result is expected since, as mentioned above, the ABCD rule was designed to recognize melanoma and not developed to recognize seborrheic keratosis. Further, we found that the color feature, defined as the number of colors appearing within the skin lesion, is not used by the classifiers. Finally, we find that no network uses the dermoscopic structures-feature. The reason for this observation is that discovering these structures is a challenging problem in itself. In a challenge that was held specifically to segment skin lesions into the dermoscopic structures, described by Codella et al. (2019), the best algorithm reached a Jaccard score of only 0.307.

Since we found that the classifiers do not use the color feature, the third group

of features quantifies the color of the skin lesion. If we include the value and the saturation in addition to the color, many of the skin lesion classifiers use it to determine their predictions.

As described earlier, one of the main concerns of domain experts against employing deep neural networks in safety or security-critical tasks like the automatic classification of skin lesions is that deep neural network classifiers might be biased. To this end, many biases have been found in different datasets of skin lesion images. Some examples are explained and investigated in Bissoto et al. (2020). For the final, fourth group of features, we focus on four biases in this work. The first two biases on which we focus are the sex and age of a patient. Some deep learning classifiers reach vastly different accuracies on images of skin lesions in women and images of skin lesions in men as well as for groups of patients of different ages, as reported for example in Muckatira (2020). Since the sex of the patient can be extracted from the image, for example, by using the body hair, the first two features in this final group are the age and the sex of the patient. The third bias we investigate is the skin color of the patient. The available datasets only contain images of people of light skin, and, hence, this test only considers different shades of light skin. We find that especially the classifier trained to recognize seborrheic keratoses use this feature. Based on these findings, we investigate the training set of the 2017 ISIC challenge dataset (Codella et al., 2018) and find an unknown bias. Finally, we investigate the existence of colorful patches in the image. These patches were introduced into the ISIC archive [2] through the introduction of the data from the Study of Nevi in Children (SONIC) Project (Scope et al., 2016). Some images of the SONIC Project contain colorful patches that are stuck to the patient's skin and can be found in the image. Since the study only considers nevi, it introduces a bias into the ISIC archive and connects colorful patches to nevi. Therefore, the colorful patches can be used to form a cleverhans predictor (Lapuschkin et al., 2019). We find that all groups of classifiers use at least one of these biases. The use of these biases demonstrates that more work is needed to train reliable and unbiased classifiers. The results of our study were presented in a talk at the ISIC-Workshop at CVPR 2021 (Reimers et al., 2021b).

The most straightforward way to train an unbiased classifier is to use an unbiased dataset. However, it is not feasible to collect an unbiased dataset for many tasks, such as the medical task described above. Collecting an unbiased dataset can be unfeasible if it is costly, dangerous or unethical to collect particular examples. Hence, we need methods to stop bias propagation from the biased dataset to the classifier. One way to achieve this goal is adversarial debiasing. The idea of adversarial debiasing is to construct and use an additional loss function that penalizes the use of a known bias feature. An essential first step in this direction is determining whether a classifier uses a specific feature. Since the method proposed in this work is a criterion for whether a classifier uses a feature, we can use it to improve adversarial debiasing if we turn the criterion into a differentiable loss function. Therefore, the second application of the method proposed in this work

[2] www.isic-archive.com

is an adversarial debiasing method in Section 4.2.2. In that section, we begin by introducing the problem of adversarial debiasing. Furthermore, we introduce the state-of-the-art and explain the differences between adversarial debiasing based on our new criterion and adversarial debiasing based on other criteria. In contrast to other methods from the literature that propose methods that fit any bias, we focus on only one, well-defined bias. To define this bias, we divide all features of an image into two categories. The first category contains features relevant to the classification task at hand, and the second category contains features that do contain no or a neglectable amount of information relevant to the classification task. An optimal classifier would utilize the features of the first category and ignore all features of the second category. However, when building a dataset, one of the second category features can contain information on the label within the dataset. This connection occurs because a feature of category two is correlated with features of category one in the finite sample. This might sometimes happen due to a lack of caution when collecting the dataset. However, it can also be due to valid concerns. One possible reason can be safety concerns. As an example, consider a situation where we collect data to train a driver assistance system that predicts the aquaplaning risk from images as proposed, for example, by Schneider et al. (2018). For the situation of low aquaplaning risk, recording images is straightforward. We can simply drive around and collect data in the wild. This procedure is not only a cheap way of generating images, but the data will, further, contain a realistic distribution of diverse road surfaces and image backgrounds that mimics the distribution we expect during the application of the system. In contrast, letting a driver drive into an aquaplaning situation is dangerous. Therefore, all images that display a high aquaplaning risk have to be recorded in a specific facility in which the driver's safety can be guaranteed, even in situations where the car starts aquaplaning. This restriction influences the distribution of street surfaces and backgrounds found in the images with high aquaplaning risk. However, the road surface and the image background are features of the second category that the system should not use to determine the aquaplaning risk during its intended application. Due to the safety risk, the classifier will link the road surface and the background in the special facility to high aquaplaning risk. Hence, the dataset is biased. Another reason why a dataset might contain a bias is ethical reasons. For example, in a medical image dataset, images of severe illness might contain artifacts caused by the treatment of the illness. The only way to create images of severe illness without these artifacts is to withhold treatment from patients that need it. Since this would be unethical, we have to accept the bias in the dataset. Finally, acquiring a specific combination of meaningful and meaningless features might be unfeasible. Specifically, if both features are rare, the combination might be missing in the dataset. These three situations demonstrate that the kind of bias tackled in this work is common in many tasks of computer vision. Further, the focus on one kind of bias allows us to give a formal definition of the creation of the bias in the form of a structural causal model. This model allows us to include a theoretical discussion, incorporating formal mathematical proof for a simple case. This proof demonstrates that the optimal classifier fulfills our

criterion of whether it uses the bias feature in a simple case. On the other hand, we prove that the optimal classifier does not fulfill the criterion used by state-of-the-art methods from the literature. To empirically evaluate the difference between these two criteria, we need to implement them as differentiable losses. Hence, in Section 4.2.2.3 we describe three different implementations of the criterion proposed in this work as a differentiable loss function. The three implementations cover the conditional mutual information, the maximum conditional correlation coefficient and the Hilbert-Schmidt conditional independence criterion. In that section, we explain the main idea of these implementations, including the relation to their counterparts in the state-of-the-art approaches from the literature.

Finally, in Section 4.3.3, we empirically investigate the difference between adversarial debiasing based on the method proposed in this work and the state-of-the-art in adversarial debiasing from the literature. To this end, we propose three experiments. For the first experiment, we create a synthetic dataset that maximizes the difference between our adversarial debiasing method and the methods from the literature. On this dataset, our method outperforms the methods from the literature. In the second experiment, we conduct an ablation study to demonstrate that the performance increases because we change how to check whether a deep neural network uses a feature. Finally, to show that these advantages of our new method also generalize to real-world images, we describe experiments on real data. On this dataset, we train a classifier that distinguishes cats from dogs. To introduce a bias into the dataset, we correlate a feature that is not suited to differentiate between cats and dogs with the meaningful features that cover the differences between cats and dogs. The meaningless feature we use for this experiment is whether the color of the animal's fur is light or dark. Using this feature, we create multiple datasets with varying amounts of bias. These datasets allow us to compare our method to a baseline classifier that does not utilize adversarial debiasing and methods from the literature on datasets with varying amounts of bias. Furthermore, these datasets allow us to evaluate the connection between the amount of bias and the improvement of our method over the state-of-the-art methods on datasets of varying amounts of bias. We observe that our method outperforms the baseline and the methods from the literature and that the improvement in accuracy on unbiased data emits a strong correlation to the amount of bias in the dataset. This study was originally published in Reimers et al. (2021a).

1.1 Adversarial Examples

In the first part of this work, we consider adversarial examples. An adversarial example is an example an adversary produces from a clean input example. The adversary's goal is to add a perturbation to the input that is imperceptible for a human observer but changes the classification of an automatic deep learning classifier.

The idea that an adversary can slightly alter an input to an automatic classification system to trick the automatic system into misclassifying the input is older than

the recent success of deep neural networks. It is discussed, for example, by Dalvi et al. (2004), and Lowd and Meek (2005) for the detection of spam in emails. Since Szegedy et al. (2013) first considered adversarial examples for deep neural networks, researchers made a great afford to find methods to create adversarial examples (Carlini and Wagner, 2017; Goodfellow et al., 2014b; Moosavi-Dezfooli et al., 2017, 2016) and to make deep neural networks more robust against adversarial attacks (Bai et al., 2017; Goodfellow et al., 2014b; Gu and Rigazio, 2014; Kannan et al., 2018; Li and Li, 2017; Lu et al., 2017; Madry et al., 2017; Papernot et al., 2016c; Rozsa et al., 2018, 2016b; Tramèr et al., 2017b).

Similar to the example of spam detection, at the beginning of the investigations into adversarial examples for deep neural networks, most of the studies focused on safety and security questions. For example, Eykholt et al. (2017), and Metzen et al. (2017) demonstrated that adversarial attacks are robust enough to fool the visual recognition systems of driver assistance systems in the real world. However, the situation in which adversarial examples are a serious security risk is not obvious. For example, Gilmer et al. (2018a) have called into question whether an actual threat model exists. They argue that more manageable and more robust attacks in almost all scenarios do not involve minimal perturbations. Hence, situations that require an adversary to find a minimal perturbation of a given, genuine input that fool a classification system but not a human observer are rare. In contrast to the research on adversarial examples prior to the interest in deep learning, the motivation for research on adversarial examples is not limited to security questions of automatic classifiers. As described above, the existence of adversarial examples proves that deep neural networks use different features than a human observer to classify an input. More specifically, visualization methods as, for example, Erhan et al. (2009) and Zeiler and Fergus (2014) suggest that the first layers of deep neural networks identify basic structures like edges or colors, which are in deeper layers combined to identify more complex parts of objects. However, adversarial examples challenge this view. A classifier that combines basic features in later layers is inherently immune to adversarial examples. The imperceptible perturbation does not change any of these basic features such as edges or corners. If later layers do nothing but combine these features, they would be immune to adversarial attacks. However, as demonstrated, for example, in Dong et al. (2017), the representations in later layers are changed by adversarial attacks. Because adversarial examples are linked to the features a deep neural network uses, researchers consider them to get a better fundamental understanding of deep neural networks. For example, Su et al. (2018) link the adversarial vulnerability of a classifier to its generalization performance, and Stutz et al. (2019) link them to the quality of gradients in interpretation tasks.

One way to understand why adversarial examples exist and, consequently, which features are selected by a deep neural network is to link a classifier's vulnerability to adversarial examples to the properties of the deep neural networks and the datasets on which they are trained. Previously, the first of these two ideas have been investigated. For example, Cubuk et al. (2017) compare nine different classifiers on the same task and compare the number of successful adversarial attacks, and Su

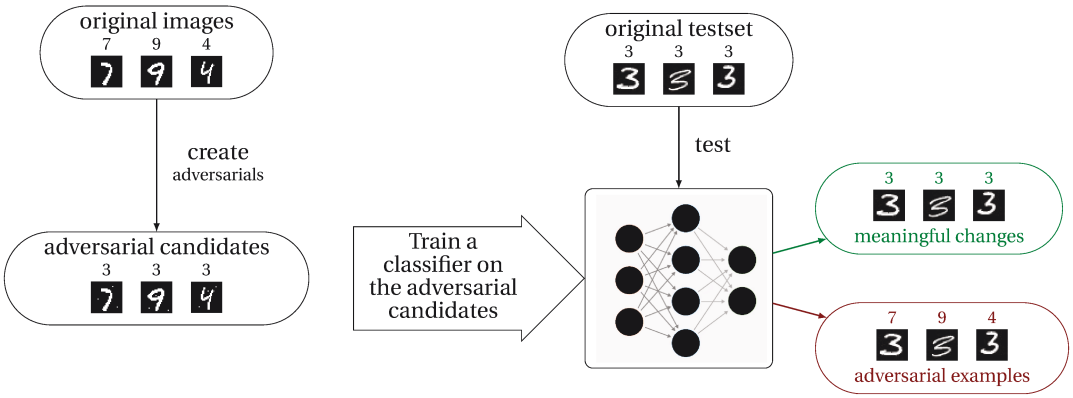


Figure 1.1: *The basic idea to distinguish between meaningful changes in images and adversarial examples. We create adversarial candidates for the images in the training set. Then, a neural network is trained to predict the adversarial labels from these candidates. The neural network is evaluated on the correctly labeled original test set. If the network is able to learn the correct relationship between the image and the label from the adversarial candidates, the candidates are not adversarial, because they resemble the target class.*

et al. (2018) and Rozsa et al. (2016a) investigate the connection between the accuracy of a classifier on clean test data and its robustness against adversarial attacks. Other researchers have compared different training methods and their influence on the robustness of a deep learning classifier against adversarial examples. They compare classifiers optimized with their new method to those optimized with the standard methods, such as stochastic gradient descent with a cross-entropy loss. Examples of papers following this idea are Goodfellow et al. (2014b), Gu and Rigazio (2014), Rozsa et al. (2016b), Papernot et al. (2016c), Lu et al. (2017), Li and Li (2017), Tramèr et al. (2017b), Madry et al. (2017), Bai et al. (2017), Kannan et al. (2018) and Rozsa et al. (2018).

The abovementioned research links classifiers and their optimization methods to their adversarial robustness. On the other hand, we link adversarial robustness to the properties of the datasets on which they are trained. In Section 3.1.2, we start by demonstrating that the properties of datasets are crucial for the vulnerability of deep neural networks to adversarial examples using two example datasets. Classifiers are vulnerable to adversarial examples when trained on one but not if trained on the other dataset, even though the datasets have the same input dimension and the same number of classes.

One of the major challenges one has to face when evaluating the adversarial robustness of a classifier is that the definition of an adversarial example depends on the notion of being indistinguishable from the clean original image. The notion of indistinguishability is linked to a human observer. Most of the works mentioned above tackle this problem by setting an arbitrary threshold on an L_p-norm, most commonly, the L_1, L_2 or L_∞-norm. However, Wang (2004) and Zhang et al. (2018b) demonstrate that neither L_p-norm is a good measure for imperceptibility. Therefore, to measure adversarial robustness, we revisit the work of Szegedy et al. (2013) that

introduced adversarial examples for deep neural networks. That work gives two important properties of the adversarial examples in addition to the definition. These two properties are that, first, we can find an adversarial example close to every example in every dataset, meaning the adversarial perturbation is small, and, second, that neither the perturbation nor the resulting adversarial image resembles the target class. The idea of how we use the latter is displayed in Figure 1.1.

These properties are the difference between the intriguing and surprising phenomenon of adversarial examples and the inevitable fact that a big enough perturbation will change the decision of a classifier. As we, for example, displayed in Figure 3.2 in Section 3.1. In Section 3, we furthermore explain how we use both of these properties to identify adversarial examples independent of the dataset. Since we can measure adversarial examples independent of the dataset, we identify a property of datasets associated with adversarial robustness, namely the multi-modality of individual classes' distribution. We find that the multi-modality of the class distributions makes the adversarial examples not resemble the target class. In our experiments in Section 3.3, the DeepFool algorithm introduced by Moosavi-Dezfooli et al. (2016) was able to find true adversarial examples in the multimodal case, while it only found adversarial examples that resemble the target class for the unimodal case.

1.2 Identifying Features Relevant for Skin Lesion Classification

The first application of our method that we describe in this work is the investigation of state-of-the-art automatic skin lesion classifiers. These classifiers are used to classify images of skin lesions. One essential task in skin lesion classification is determining whether a skin lesion is skin cancer.

Skin cancer is one of the most common forms of cancer, and melanoma is the most dangerous form of skin cancer. As described, for example, by Geller et al. (2007), the most promising method to increase the chance of survival for patients is early diagnosis. Regular checkups by trained medical professionals are needed to guarantee an early diagnosis. However, the medical professionals necessary to offer such labor-intensive examinations comprehensively are not available in many regions. One possibility to reduce the amount of human labor necessary is to employ automatic skin lesion classifiers.

To this end, many methods have been proposed to support practitioners. In one approach that, for example, Celebi et al. (2007) pursue, hand-crafted features are automatically extracted from the input and then processed by a simple classification method. Another approach is to use deep neural networks. For example, Perez et al. (2018),Gessert et al. (2020), Esteva et al. (2017) and Brinker et al. (2019) feed the sample images directly into a deep neural network, which performs feature extraction and classification in a single step.

The main advantage of the former approach is interpretability. Since the classifi-

cation methods used in this approach are simple, the main difficulty is selecting features. One suitable set of features used in these algorithms is the set of features named in the dermoscopic ABCD rule presented by Nachbar et al. (1994) and Stolz and Kunz (2021). The ABCD rule is an algorithm for dermatologists to differentiate between melanoma and nevi in dermoscopic skin lesion images. To aid with this differentiation, the user extracts four features of the skin lesion, namely **A**symmetry, **B**order, **C**olor and **D**ermoscopic structures. A numerical value for each feature is calculated, and all values are combined linearly into a total dermoscopy score. Thresholding this total score yields high accuracy to distinguish melanoma from benign nevi. When using these features, we can guarantee that the automatic classifier bases its decision on meaningful features and, therefore, dermatologists and patients can trust the predictions of these algorithms. The fact that the ABCD rule is an algorithm makes it straightforward to automatize. The most challenging task is to extract the four abovementioned features automatically.

In contrast, the deep learning approach has the advantage of higher accuracy. In recent years, a new state-of-the-art has formed. Instead of extracting hand-crafted features according to the ABCD rule, for example, in Gessert et al. (2020) a large ensemble of very deep neural networks selects the features automatically. Additionally, pretraining on large out of domain image datasets like ImageNet (Russakovsky et al., 2013) and heavy data augmentation, including test time augmentation, as proposed in Perez et al. (2018) is used. These advancements allow researchers to create automatic skin lesion classification systems that outperformed even experienced practitioners, as was evaluated by Brinker et al. (2019) and Tschandl et al. (2019). However, a user who employs a deep learning system has no control over the feature selection process of the system's features for classification, because, as described, for example, by Reimers and Requena-Mesa (2020) a central idea of deep learning is automatic feature extraction. In the automatic skin lesion classification setting, it is challenging to determine whether the classifier still relies on the features named in the ABCD rule. Instead classifiers might rely on bias features, which are, for example, described by Mishra and Celebi (2016), Rieger et al. (2020), Muckatira (2020) and Bissoto et al. (2019) and exist in all datasets.

To tackle this question, in Section 4.2.1 we present three results. First, we verify that the method presented in this work is suitable for classifiers in the complex, real-world problem of automatic skin lesion classification. To this end, we selected four features of skin lesion images that contain little to no information useful towards classifying the skin lesion in the image. Our experiments show that our method produces almost no false positives for these meaningless features. Second, we investigate whether state-of-the-art deep neural networks use the features named in the ABCD rule. Our experiments show that networks trained to identify melanoma use the asymmetry and border feature but not the color and dermoscopic structure feature. In contrast, models trained to identify seborrheic keratosis use only the color, but neither asymmetry and border nor the dermoscopic structures. Third, we test whether the deep neural networks rely on known biases. Our experiments find that the classifiers use the patients' age and skin color to classify a skin lesion. Fur-

ther, we find that classifiers use the spurious connection between colorful patches and nevi in the SONIC dataset (Scope et al., 2016) which Mishra and Celebi (2016) and Rieger et al. (2020) previously reported. We describe the exact features and how we score them automatically in Section 4.2.1.4. Furthermore, we describe the experimental results in Section 4.3.2.

1.3 Debiasing

The second application we use to investigate the performance of the method described in this work is adversarial debiasing.

The impressive performance of deep neural networks makes their use in many areas more desirable. These areas comprise classical computer vision tasks like object detection, for example, Russakovsky et al. (2015) and semantic segmentation, for example, Long et al. (2015). Furthermore, they include safety- and security-critical areas, for example, the prediction of recidivism described in Angwin et al. (2016) or medical tasks, for example, the automatic classification of skin lesions described in the previous section and presented, for example, in Perez et al. (2018), Gessert et al. (2020) and Tschandl et al. (2018). However, many domain experts have concerns about using automatic deep learning classifiers, especially for safety and security-critical tasks. Even though, for example, Tschandl et al. (2019) shows that these classifiers outperform human experts, users fear biased classifiers. Indeed, for example, in the task of automatic skin lesion classification Muckatira (2020) show that the performance of classifiers varies across age groups, and Wang et al. (2020) demonstrates that many image datasets contain biases.

One main reason classifiers are biased is that they are trained on biased datasets. Every dataset is a unique slice through the visual world (Torralba and Efros, 2011). Hence, a dataset does not represent the real world perfectly. To further describe the bias, we partition all input features into two sets. The first set contains features relevant to the classification task, and the second set contains all features that are not relevant for the classification task. The optimal classifier identifies the first set of features and uses it to predict the class. However, a feature from the second group is correlated to a class label in a biased dataset. We call such a feature a bias feature. As described above, this is not necessarily due to carelessness during the data collection process but might instead be due to security or safety concerns, due to ethical considerations or due to it being difficult to acquire certain samples. If a classifier is trained on such a dataset, it might use this meaningless feature to create a “Clever-Hans” classifier (Lapuschkin et al., 2019). One way to stop a classifier from picking up such a bias-feature is adversarial debiasing. The idea of adversarial debiasing is to introduce a second loss function $\mathcal{L}_{db}$ in addition to the classification loss $\mathcal{L}_{cl}$. This second loss function, which we call the debiasing loss, penalizes the use of the feature by the classifier.

The first step in this direction is a quantitative way to measure whether the classifier uses a feature. To this end, related work (see Section 4.2.1.2) from the literature uses statistical dependence between the bias feature B and an intermediate

representation R extracted from the deep neural network

$$B \perp\!\!\!\perp R. \tag{1.1}$$

In the main part of this work, in Section 4.1, we demonstrate that this criterion is too strict. While a representation can only be independent of a bias feature if the classifier does not use it, the inverse is false. Even if a classifier ignores a feature, the independence in (1.1) might still not hold. Instead, we use the conditional dependence, conditioned on the ground truth label L,

$$B \perp\!\!\!\perp R \,|\, L. \tag{1.2}$$

As discussed, for example, in Wang et al. (2020), various kinds of bias exist for a multitude of reasons. In contrast to other papers from the literature, we do not propose our method as a solution for every kind of bias. In contrast, we tackle only one kind of bias for which we present a specific bias model in Section 4.1.2. This formal description of the data generation model has two advantages. First, the formal description allows us to provide rigorous mathematical proof. To this end, we prove that the optimal classifier fulfills our conditional independence criterion (1.2) but not the independence criterion (1.1). Second, this specific bias model allows users to determine whether the method is suitable for a given situation.

To evaluate the change in criterion empirically, we need to turn the conditional independence test into a differentiable loss. To this end, we propose to use the test statistic of various conditional dependence tests, namely the conditional mutual information (Wyner, 1978, Lemma 3.1), the maximum partial correlation coefficient (Sarmanov, 1958) and the Hilbert-Schmidt conditional dependence criterion (Gretton et al., 2007). We provide further explanations in Section 4.2.2.3. We demonstrate that these new loss functions lead to a higher accuracy on an unbiased test set in Section 4.3.3. In Section 4.3.3.1, we observe this increased accuracy in experiments on a synthetic dataset, in Section 4.3.3.2, in experiments that show that this increase in accuracy is due to the change in the criterion and, in Section 4.3.3.3, experiments that show that this increase in accuracy generalizes to real-world images.

1.4 Outline

The rest of this work is structured as follows: We start, in Section 2, by introducing the necessary fundamentals that are needed to follow the rest of this work. The first of these fundamentels is the structural causal model theory introduced by Pearl (2009) and Peters et al. (2017) in Section 2.1. To this end, we motivate causal modeling in Section 2.1.3, introduce the framework of Pearl (2009) in Section 2.1.2, discussing some shortcomings and limitations of this framework in Section 2.1.3 and, finally, discuss why this framework is a good fit for the problem tackled in this work in Section 2.1.4. Afterward, in Section 2.2, we introduce the basic concepts of deep learning. We begin that section by describing the problem of statistical learn-

ing and, in particular, the problem of deep learning in Section 2.2.1 and, afterward, explain why neural networks are a suitable instrument to tackle deep learning tasks in Section 2.2.2. The method we propose in this work reduces whether a deep neural network uses a feature to a conditional dependence test. Hence, we introduce the theory of dependence tests as well as three unconditional and three conditional dependence tests in Section 2.3. These dependence tests include the correlation and partial correlation criterion in Section 2.3.1, the maximum correlation coefficient criterion and the maximum partial correlation coefficient criterion in Section 2.3.2, the mutual information criterion and conditional mutual information criterion in Section 2.3.3 and, finally, the Hilbert-Schmidt independence criterion and the Hilbert-Schmidt conditional independence criterion in Section 2.3.4. Afterward, in Section 2.4, we introduce the state-of-the-art in understanding which features a deep neural network uses. Our primary focus in this section lies on saliency maps. We introduce the general idea of saliency map methods in Section 2.4.1. This introduction includes the three main ways to create saliency maps, namely gradient-based methods in Section 2.4.1.1, methods based on obfuscation of parts in the input image in Section 2.4.1.2 and methods based on a combination of the value and the gradient of the neural network at the input image in Section 2.4.1.3. Afterward, we discuss other methods not based on saliency maps in Section 2.4.2. We describe the method using concept activation vectors that is presented in Kim et al. (2018) in Section 2.4.2.1 and the causal concept effect method presented in Goyal et al. (2019) in Section 2.4.2.2. We explain methods that link intermediate representations of a deep neural network to semantic concepts in Section 2.4.2.3 and, finally, methods that find images that maximize the score for a specific class, in Section 2.4.2.4, and methods that explain the classifier using examples from the training data in Section 2.4.2.5. As discussed above, we use adversarial examples to demonstrate the lack of knowledge about which feature a deep neural network extracts from the training set and uses for its predictions. Therefore, we close the section on basics with an introduction to adversarial examples in Section 2.5. This section includes an overview on methods to create adversarial examples in Section 2.5.1. It contains, in Section 2.5.2, a discussion of the most intriguing properties of adversarial examples, namely, that they exist close to each clean examples in Section 2.5.2.1, that they do not resemble the target class in Section 2.5.2.2, that they are robust to random noise in Section 2.5.2.3, that adversarial examples are transferable between classifiers in Section 2.5.2.4, that the curvature of the decision boundary is positive close to them in Section 2.5.2.5 and, finally, that the robustness and the accuracy of a classifier are related in Section 2.5.2.6.

Further, in Section 2.5.3, we discuss whether adversarial examples provide a realistic threat in real-world situations. Finally, in Section 2.5.4, we list suggestions from the literature on why adversarial examples exist. These suggestions include the theory of pockets of low probability in Section 2.5.4.1, the theory of overfitting a low dimensional manifold in Section 2.5.4.2, the theory that the effect of correctly classified examples is too small in Section 2.5.4.3, the theory that the neural networks are too linear in Section 2.5.4.4, the theory that adversarial examples are a natural

consequence of misclassification in Section 2.5.4.5 and the theory of boundary tilting in Section 2.5.4.6.

We then, in Section 3 describe our work in the field of adversarial examples. Our main contribution in this field is to find an objective measure for the defining principle of imperceptibility independent of the dataset. This objective measure allows us to evaluate the influence of properties of the datasets on the adversarial robustness of classifiers. We start the section by explaining the drawbacks of the state-of-the-art ways to define adversarial examples and show why they do not generalize across datasets in Section 3.1. We, afterward, describe how we adapt this definition to make it comparable across datasets in Section 3.2. In Section 3.3 we present three experiments. First, in Section 3.3.1, we demonstrate that the definitions we proposed in this work is dataset agnostic. Second, we compare the adversarial robustness measured with the different definitions of adversarial examples in Section 3.3.2. In the third experiment, in Section 3.3.3, we demonstrate how these new definitions are used to find the relation between properties of the dataset and the adversarial robustness of the classifier by demonstrating that the adversarial robustness of classifiers trained on datasets where the distributions of individual classes are multimodal is less than the adversarial robustness on classifiers trained on other datasets.

We present the main contribution of our work in Section 4. That section is divided into three parts. The first part, Section 4.1, introduces the theoretical background of our new method to determine whether a deep neural network uses a feature. We start this section with an illustrative example in Section 4.1.1 and continue with the explanation of the structural causal model that we construct to represent the setting of supervised learning in Section 4.1.2. We first introduce all variables and processes that are part of supervised learning and describe how we can combine them in a structural causal model in Section 4.1.3. Afterward, we present an introductory example to illustrate the described variables in Section 4.1.4. Further, we describe the structural causal model and focus on the scope and limitations of our proposed method. First, in Section 4.1.5, we discuss examples of supervised learning where the introduced structural causal model does not fit. In Section 4.1.5.1, we discuss situations in which the inference function has different inputs, and, in Section 4.1.5.2, we discuss the possibility that one of the involved processes is deterministic. We find that in both these cases, the conclusions that we drew in Section 4.1.2 hold. In contrast, we describe the limitations of our method in Section 4.1.6. In Section 4.1.7, we describe the relations between our new method and the state-of-the-art methods from the literature. In Section 4.1.7.1 we compare it to the saliency map methods. Furthermore, in Section 4.1.7.2 we compare it with the method using concept activation vectors that is presented in Kim et al. (2018) and in Section 4.1.7.3 to the causal concept effect method presented in Goyal et al. (2019). We, furthermore, compare our method to methods that also rely on causal modeling but use the neural network as a structural causal model in Section 4.1.7.4 and, in Section 4.1.7.5, to methods of feature visualization. Finally, in Section 4.1.7.6, we compare it to methods that explain the classifier using examples

from the training data.

Following these theoretical considerations, we discuss the two main applications in Section 4.2. First, we present the direct application of determining which features are used by state-of-the-art automatic skin lesion classification systems in Section 4.2.1 and, second, in Section 4.2.2, the application of our method to the problem of adversarial debiasing. We start Section 4.2.1, the section concerning the application to skin lesion classification, by arguing why the method presented in this work is suitable for this challanging real-live task in Section 4.2.1.1. In Section 4.2.1.2, we discuss the related work from the literature. We introduce the classifiers we investigate in Section 4.2.1.3 and the four groups of features in Section 4.2.1.4. In the second application, the application to adversarial debiasing presented in Section 4.2.2, we start by introducing the problem of adversarial debiasing in Section 4.2.2.1. As explained in the introduction, we tackle a specific bias. To define this bias, we introduce the data creation model, a structural causal model, in Section 4.2.2.2. In that section, we further present a discussion on the situations in machine learning where we think that the bias model is suitable and for which situations we do not consider it suitable. Further, we include proof that demonstrates that, in a simple setting, the optimal classifier will fulfill our proposed criterion, while it will not fulfill the criterion, which is more widely used throughout the literature. Finally, in Section 4.2.2.3, we present three ways to implement our new criterion as a differentiable loss function.

In Section 4.3, we present the empirical evidence that supports the theoretical statements in the earlier sections. That section is divided into three parts. In the first part, Section 4.3.1, we present the experiments to corroborate the general useability of our method on complex, real-world datasets and show that it returns correct and reasonable results. In the second part, in Section 4.3.2, we present the results for which feature is used by state-of-the-art automatic skin lesion classification methods. Finally, in the third part, in Section 4.3.3, we present evidence that underpins the claim that our new criterion outperforms the methods from the literature in adversarial debiasing if the bias was created as presented in Section 4.2.2.2. The first part contains three experiments. First, in Section 4.3.1.1, we present an experiment on synthetic data created to verify that our method produces the correct results. The second experiment, which we present in Seciton 4.3.1.2, demonstrates how this method can be used to compare classifiers on specific aspects of the data without using a specialized dataset. Finally, in Section 4.3.1.3, we present a further experiment on the fine-grained problem of distinguishing bird species. The third part of Section 4.3, the experiments to underpin the use of our method as an adversarial debiasing criterion, also starts with a synthetic data experiment designed to maximize the difference between the two criteria in Section 4.3.3.1. In Section 4.3.3.2, we present an ablation study that shows that we can attribute the increase in accuracy to the change in criterion. Finally, we demonstrate that the increase in accuracy can also be observed in real-world datasets in Section 4.3.3.3.

We finish the work by summarizing our conclusions in Section 5 and describing possible directions of future work in Section 6.

2 | Background and Basics

In this section, we introduce some basic concepts and frameworks needed to understand the rest of this work. The main focus of this work is whether a feature causes or, in other words, is relevant for the prediction of a deep neural network. Therefore, as the first concept, in Section 2.1, we introduce the notation of cause and effect, particularly, the causal modeling framework introduced in Pearl (2009) and Peters et al. (2017). Afterward, in Section 2.2, we present the problem of statistical learning, including some applications and deep neural networks as a suitable solution for these applications. A major component of the method proposed in this work is statistical dependence tests. Section 2.3 introduces the concept of statistical dependence tests and describes well-known dependence tests, namely correlation and partial correlation, maximum correlation coefficient and maximum partial correlation coefficient, mutual information and conditional mutual information, and the Hilbert-Schmidt independence criterion and the Hilbert Schmidt conditional independence criterion. Further, in Section 2.4, we introduce state-of-the-art in feature attribution and visualization of deep neural networks. Finally, in Section 2.5, we introduce the concept of adversarial examples.

2.1 Cause-Effect

This section introduces the basic concepts of the causal modeling framework introduced by Pearl (2009) and Peters et al. (2017). We first, in Section 2.1.1, introduce the problem that causal modeling is aiming to solve. Afterward, in Section 2.1.2, we explain the basics of this causal modeling framework. Finally, we conclude with some drawbacks of this causal modeling framework, with a comparison to other causality frameworks and with a short conclusion in Section 2.1.3, as well as a discussison on why this framework is suitable for the problem tackeled in this work in Section 2.1.4.

2.1.1 Motivation of Causal Inference

The goal of statistical methods is to describe and analyze data. If we observe a system composed of multiple variables $X_1, \ldots, X_n$, all information about this

system is in the joint distribution

$$\mathbb{P}(X_1, \ldots, X_n) \tag{2.1}$$

of the variables involved. All other distributions, such as the marginal distribution of one variable

$$\mathbb{P}(X_1) = \int \mathbb{P}(X_1, \ldots, X_n) dx_2 \ldots x_n \tag{2.2}$$

or the conditional distribution

$$\mathbb{P}(X_1 \,|\, X_2, \ldots, X_n) = \frac{\mathbb{P}(X_1, \ldots, X_n)}{\mathbb{P}(X_2, \ldots, X_n)} \tag{2.3}$$

can be calculated from the joint distribution. These distributions allow us to solve many interesting problems in machine learning. For example, if we want to identify outliers in a dataset, we can use the joint distribution to find the likelihood for each example. Further, we can use the conditional distribution to, for example, fill gaps in data where one of the variables is unknown while the other variables are observed. Finally, knowing the joint distribution allows us to sample more unknown examples from the same distribution.

However, there are also some tasks we can not solve, even if we have access to the joint distribution. One of these tasks is to predict how a system will react if the distribution of one of the involved variables changes. To illustrate this fact, we provide the following example. This example involves two systems, each containing two variables, X and Y. In the first system, these are given by

$$X_1 \sim \mathcal{N}(0, 1) \tag{2.4}$$

$$\varepsilon \sim \mathcal{N}(0, 1) \tag{2.5}$$

$$Y_1 \nleftarrow \frac{1}{\sqrt{2}} X_1 + \frac{1}{\sqrt{2}} \varepsilon. \tag{2.6}$$

In contrast, in the second example, these variables are connected by

$$Y_2 \sim \mathcal{N}(0, 1) \tag{2.7}$$

$$\varepsilon \sim \mathcal{N}(0, 1) \tag{2.8}$$

$$X_2 \nleftarrow \frac{1}{\sqrt{2}} Y_2 + \frac{1}{\sqrt{2}} \varepsilon. \tag{2.9}$$

Here, the $\nleftarrow$ indicates, that a change in the right hand side would cause a change in the left hand side. In contrast, a change in the left hand side will not result in a change in the right hand side. Both of these systems lead to the same joint distribution. In both systems, the two variables follow the distribution

$$X_i, Y_i = \mathcal{N}\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 & 0.5 \\ 0.5 & 1 \end{pmatrix} \right). \tag{2.10}$$

However, if the distribution of either X or Y changes, these two systems respond in a very different way. When changing X in the first system, the distribution of Y will also change. In contrast, if we change the distribution of X in the second system, the distribution of Y will stay the same. If we are presented with one of these systems at random, we can not know which of the systems it is because the joint distribution of both systems is identical under the initial conditions.

To formalize these systems, Pearl (2009) introduces a hierarchy of questions, which they call the "ladder of causality." According to this ladder, questions about systems can be categorized into three ranks. The lowest rank is the rank of "association." This rank contains all questions that can be answered from observations. Examples for these kinds of problems are, as mentioned above, outlier detection, anomaly detection, classification, regression and generating new examples. The advantage of this lowest rank is that the answers to questions of this rank can be validated using observations. Hence, these questions are the only questions for which we can validate findings for systems that we can only observe but not manipulate as, for example, astrophysical systems.

The second rank of the ladder of causality is called the rank of "intervention." This rank contains exactly the abovementioned kind of questions that can not be solved from the joint distribution alone. Questions of this rank often concern how the system will respond if the distribution of one of the variables changes or which variables' distribution change if we manipulate the distribution of one variable in the system. Since changing one of the variables in a system away from its observed distribution can be understood as an intervention on the system, this rank is called "intervention." The answers to questions on this rank can not be verified from observations. The only way to answer them is to manipulate the variable in the system to follow the prescribed distribution without interfering with the other variables. This is, however, only possible for some systems, and even in systems in which we can intervene, setting a variable to a specific intervention distribution might prove impossible.

The final and third rung of the ladder of causality is the rank of "counterfactuals." As the name suggests, counterfactuals are data points that did not happen. Hence these data points are not factual but counterfactual. Questions on this rank are of the form "What would have happened if A would have happened instead of B?" The difference between a counterfactual and an intervention is that all random variables involved are sampled again from the new intervention distribution when conducting an intervention. In contrast, when considering a counterfactual, we do not resample any of the variables but assume that they stay on their original value, not considering how likely or unlikely this value is under the new intervention distribution. We use the following example to illustrate the difference between intervention and counterfactuals further. The example system is a game where a player tries to predict a dice-roll of a fair six-sided dice. The system contains three variables. These variables are the number the player guesses, the number the dice shows after it is rolled and the binary variable indicating whether the player has won or not. Let us assume we observe one run of this game and find

that the player guesses a four, the dice then shows a five and, consequently, the player has lost the game. An example question on the first rank of the causal ladder would be to determine whether the player has won given his guess and the dice roll result. An example for a question on the rank of intervention would ask whether the chance of winning for the player would change if we force them to guess five every game. The answer to this question is negative because the chance of winning is 0.167, independent of the player's guess. In contrast, the counterfactual question is: "Would the player have won if they had guessed a five in this run of the game?" The answer to this question is yes. This discrepancy demonstrates the difference between questions of rank two and questions of rank three. However, it also reflects a drawback. We can often verify the answers to questions of rank two by conducting experiments and manipulating the system. In comparison, counterfactual questions can not be answered since they consider past events that we can no longer control. Since the player makes their guess before the dice is thrown, we can not go back in time and manipulate it afterward. If we, furthermore, manipulate it and throw the dice again, the dice might show a different number.

Even though questions above the first rank are hard to verify, many scientific questions are in these categories. For example, questions that ask how the Earth's climate will react to increasing the amount of atmospheric CO_2 are on the second rank of the ladder of causality. Additionally, if we want to understand the effect of an event, such as a natural disaster, we often try to estimate how the environment would look if the natural disaster would not have occurred. This question is counterfactual.

Hence, researchers have spent a lot of time and energy to find ways to answer, or at least talk about, questions on the second or third rank of the ladder of causality. In the next section, we introduce one of these solutions. Specifically, we introduce the causal modeling framework using structural causal models introduced by Pearl (2009).

2.1.2 The Framework of Pearl (2009)

The main part of the causal modeling framework described in Pearl (2009) and Peters et al. (2017) is the structural causal model (SCM). A structural causal model consists of three parts. A set of endogenous variables $\{X_1, \dots X_n\}$, a set of exogenous variables $\{\varepsilon_1, \dots, \varepsilon_n\}$ and a set of functions $\{f_1, \dots, f_n\}$ connecting these variables. All three sets have the same number of elements. The exogenous variables are pairwise independent. Each endogenous variable X is calculated by one of the functions using one of the exogenous variables and some of the other endogenous variables. These other variables are called the "parents" of the variable X. We denote them by $\mathcal{P}(X)$. Similarly, we call X a "descendant" of its parents.

We can represent a causal structural model as a directed graph by using the endogenous variables as the nodes. In this graph, an edge from node X_i to X_j exists, if and only if X_i is a parent of X_j. One of the main assumptions in the framework of Pearl (2009) is that this directed graph is circle free. This assumption corresponds

to the assumption that no variable influences itself in the modeled system, neither directly nor indirectly. One conclusion of this assumption is that we can number the nodes or endogenous variables of the system so that, for every i, all parents of the variable X_i have an index smaller than i. The corresponding SCM can be denoted as

$$X_1 = f_1(\varepsilon_1) \tag{2.11}$$
$$X_2 = f_2(\mathcal{P}(X_2), \varepsilon_2) \tag{2.12}$$
$$\vdots \tag{2.13}$$
$$X_{n-1} = f_{n-1}(\mathcal{P}(X_{n-1}), \varepsilon_{n-1}) \tag{2.14}$$
$$X_n = f_n(\mathcal{P}(X_n), \varepsilon_n). \tag{2.15}$$

As an example, we describe how to model the game described above. The game contains three endogenous variables, namely G, the player's guess, D, the result of the dice throw, and W, which has a value of one if the player has won and zero otherwise. The exogenous variables are ε_G which determines with which probability the player is guessing each number, ε_D, which is the probability for each face of the dice. more specifically, it is 0.167 for each number between one and six, and ε_W, which is trivial as W is fully determined by G and D and has no internal variability. The structural causal model describing this system is then given by

$$G = \varepsilon_G \tag{2.16}$$
$$D = \varepsilon_D \tag{2.17}$$
$$W = \begin{cases} 1 & \text{if } D = G \\ 0 & \text{else} \end{cases}. \tag{2.18}$$

Here the functions f_G and f_D are both the identity, while f_W is the characteristic function of $\{D, G, \varepsilon_W \mid D = G\}$, which is independent of ε_W. This structural causal model is represented by the graphical model displayed in Figure 2.1.

One of the great advantages of representing a system by a structural causal model is that it lets us understand which pairs of variables are dependent and which are independent. To this end, it is essential to understand which path through the causal models will lead to mutual information between variables. The first observation is that such a path does not need to be directed. A simple counterexample is the dependence between shoe size and handwriting ability in children. These are dependent, even though neither one influences the other, but the age of the children drives both. The corresponding graphical model is *Handwriting ability* $\leftarrow$ *Age* $\rightarrow$ *Shoesize* with no directed path between the two. More generally, the nodes along a path can be categorized into three categories. The first category are nodes through which the path is directed, namely "$\ldots \rightarrow X \rightarrow \ldots$" or "$\ldots \leftarrow X \leftarrow \ldots$" The second class of nodes contains nodes that influence both neighboring variables along the path, meaning a node of the form "$\ldots \leftarrow X \rightarrow \ldots$" The variable corresponding to such a node is called a "confounder." An example of a confounder is the *Age*

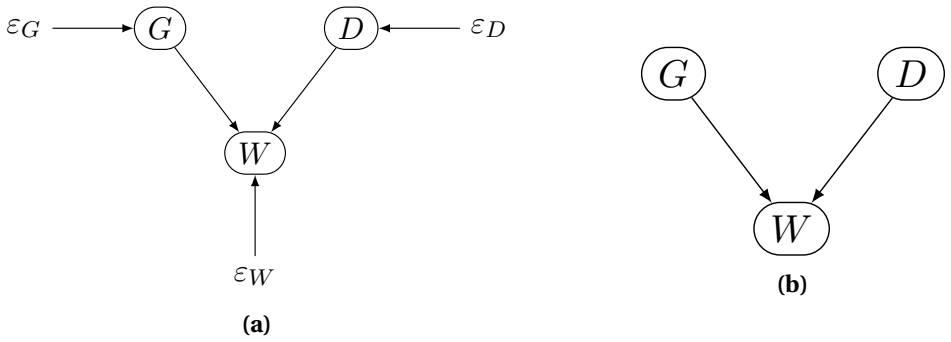


Figure 2.1: *The structural causal model for the dice guessing game. The endogenous variables G, D and W form the nodes of the model, and the the arrows between them represent the functions f_W in graph **(a)**. We also included the exogenous variables $\varepsilon_G, \varepsilon_D, \varepsilon_W$. However, in the usual graphical representation of an SCM, as displayed in **(b)**, the exogenous variables are omitted.*

variable in the example above, which is a confounder of whildren's shoe size and handwriting ability. The third class of nodes contains nodes which are influenced by both, the predecessor along the path and the successor along the path, omitting the form "$\ldots \to X \leftarrow \ldots$" The variables associated with this kind of node are called "colliders." An example of such a collider in the dice game is the variable W, which indicates whether the player has won.

If we now consider an undirected path, we say it is open if it does not contain a collider. Further, we can close an open path if we condition on a non-collider variable along it. Consider the age variable in the example above. If we condition on the age, meaning considering only children of the same age, we expect to find no statistical dependence between the shoe size and the handwriting skills. The path through the age variable would be closed. Similar to closing a path by conditioning on a non-collider variable, we can also open a path by conditioning on all colliders along it. As an example, consider the dice game. If we condition on W, meaning considering, for example, only realizations in which the player won the game, we will find a strong statistical dependence between the previously independent guess G of the player and result D of the dice throw. If we know that the player won and guessed a five, we can deduct that the dice must have shown a five. The same is true if we condition on a descendant of a collider or a non-collider (Pearl, 2009; Peters et al., 2017).

Situations like these can be modeled using a graphical model. This graphical model should have an unblocked path between two variables if and only if they are statistically dependent. To ensure this, we employ two more assumptions. The first assumption is related to Reichenbach's common cause principle. This principle is introduced in Reichenbach (1991) and states that if two variables are correlated, either one of them is causing the other or there exists a third variable that causes both of them. We use an updated version of this principle for the framework of causal modeling. First, we replace the correlation by statistical dependence to take non-linear relations into account. Second, as discussed above, in addition to the

possibilities that one variable is causing the other and that a third variable is causing both variables, we consider the third possibility that both variables cause a collider variable on which we condition.

Less formally, the central claim of this assumption is that variables can not be correlated or statistically dependent without reason. Hence, whenever we detect a statistical dependence, there has to be an open path connecting the two variables in the corresponding graphical model. Since this assumption was introduced in Reichenbach (1991), a lot of literature that discusses the philosophical perspective of this assumption exists. The assumption is widely accepted for most situations with some exceptions, for example, the situation described in Bell (1964).

The second assumption is much stronger and less discussed in the philosophical literature. This assumption states that if a variable is causing another, then the distributions of the two variables are statistically dependent. Using this second assumption ensures the other direction, namely that if an open path between two variables exists, then the two variables will be dependent.

If we accept these assumptions, we can use the resulting causal structural models to answer questions on the second and third ranks of the ladder of causality. As described above, the association between variables is different from the effect a variable has on another. While the association is through any open path, the actual causal influence from a variable X_1 to a variable X_2 is evaluated only along directed paths from X_1 to X_2. To assess the causal effect, we can, therefore, close all open paths that are not directed paths from X_1 to X_2 by conditioning on variables along them. Then, a straightforward regression will correctly evaluate the causal influence of X_1 on X_2.

One of the great advantages of structural causal models is that a rigorous mathematical theory exists inside the model. This theory allows us to test individual causal relations directly using independences tests and conditional independence tests in an automated fashion Runge (2020); Runge et al. (2019). Further, a lot of additional research has broadened the applicability of the model, for example, by allowing for known but unobserved variables Gerhardus and Runge (2020).

2.1.3 Criticism and Comparison to Other Causality Frameworks

This section, first introduces other frameworks of causality and, afterward, explains the differences to the framework introduced by Pearl (2009).

Since the questions that are on the second and third ranks of the ladder of causality are essential in many fields of science, multiple methods to answer these questions were proposed, for example, by Berkeley (1881); Granger (1969); Hill (1965); Hume (1896); Mackie (1965).

These frameworks can be divided into two philosophical definitions of causality: Generative causality and Regularity causality as discussed, for example, in Thygesen et al. (2005).

The generative view of causality sees a causal link from a variable X_1 to a variable

X_2 if X_1 influences X_2 through some physical process or mechanism. However, it is often unclear what constitutes a mechanism or process, as described in Dalkin et al. (2015). It is unclear how to evaluate or verify whether a physical process exists. The extreme form of this definition of causality is the definition that is, for example, presented in Mackie (1965), stating that an event X_1 is causing an event X_2 if and only if X_1 is necessary and sufficient for event X_2 to happen. This definition is very restrictive. If we consider, for example, the relation between smoking and cancer, we find that smoking is neither necessary nor sufficient for someone to get cancer.

The alternative is the regularity definition of causality. For example, Hume (1896) and Berkeley (1881) define a cause and effect by the cause being followed by the effect, and every object similar to the cause will be followed by an object similar to the effect. Hence, this definition is based on statistical relations of the appearance of events and less on processes. One of the frameworks that follow this view is the framework presented in Granger (1969). This framework for causality considers more relations as causal than other frameworks. More specifically, this framework defines a variable X_1 to be a cause of variable X_2 if X_1 is measured before X_2 and the value of X_1 is useful to predict the value of X_2 even if the past of X_2 is known. If we consider the example of shoe size and handwriting ability in children, this framework might detect a causal link between the two because knowing a child's shoe size will help to predict its hand writing ability.

In summary, following the generative view on causality leads to causality that can not be detected from data and seems to be a too strict criterion for causality. In contrast, following the regularity view on causality leads to identifying connections as causal, which we usually would not consider being causal. These problems have discouraged some researchers from using causality as a scientific concept. For example, Russell (2013) stated:

> "The law of causality, I believe, like much that passes muster among philosophers, is a relic of a bygone age, surviving, like the monarchy, only because it is erroneously supposed to do no harm." (Russell, 2013)

Other researchers think that this necessitates a compromise between the two criteria. The most widely accepted compromise and the definition that most closely matches the colloquial view of causality is the framework proposed by Hill (1965). That paper introduces nine heuristic criteria to decide whether a specific dependency is due to a causal relation case-by-case. These criteria are neither necessary nor sufficient to decide that a relationship is causal, but they can help a researcher make a sensible case-by-case decision. The criteria are:

1. **Strength:** The first criterium is the strength of the statistical dependence. A stronger connection, meaning that the potential cause can explain more of the variance in the potential effect makes it more likely that the relation should be considered causal.

2. **Consistency:** The second criterium demands that the relation between the potential cause and the potential effect is visible in a various contexts. This

consistency reduces the chance that the real cause of the potential effect can be found in the context.

3. **Specificity:** The more specific the potential cause and the potential effect are, the harder it is to fulfill the other criteria by chance.

4. **Temporality:** One of Hill's criteria is that the potential cause should appear before the potential effect. This criterium is widely viewed more like a criterium to decide on the direction of the causal link and less as a criterium to identify whether a relationship should be considered causal.

5. **Biological gradient:** As an epidemiologist, Hill focused mainly on biological processes. However, the main idea can be adapted to most other fields of science. This main idea is that a stronger cause should lead to a stronger effect.

6. **Plausibility:** If a causal relation is more plausible by prior knowledge, we need less justification from the data to accept its existence.

7. **Coherence:** The criterium of coherence is very similar to the criterium of plausibility. The focus for this criterium is on not contradicting any prior knowledge.

8. **Experiment and Analogy:** This criterium is similar to the two above. The focus of this criterium is more on comparing to related fields. In Hill (1965), experiment and analogy are counted as two criteria.

These criteria are a combination of both the regularity and the generative view of causality (Thygesen et al., 2005) but are neither meant nor suitable as a hard criterion but only to support a researcher to make a case-by-case decision. The criteria form three groups: The statistical criteria are the strength, consistency and biological gradient. The semantic criteria namely, specificity, plausibility, coherence, experiment and analogy. The temporality criterion forms its own group. The first group corresponds to the regularity view on causality, and the second group corresponds to a generative view on causality. The temporality criterion is used to orient the causal link.

We consider a scale from the most restrictive framework of causality (Mackie, 1965) to the framework that considers most links causal (Granger, 1969). The framework of Pearl (2009) is closer to Granger (1969) than the criteria proposed by Hill (1965). The main reason is that the framework of Pearl (2009) is similar to other frameworks like, for example, San Liang (2014), based only on data and does not consider semantic features. While this allows automatic and efficient detection of causal mechanisms under certain assumptions, it also has some drawbacks and fail-cases.

Reichenbach's common cause principle (Reichenbach, 1991) is discussed in the literature and widely accepted outside of quantum effects (Bell, 1964). The

inverse, meaning the assumption that a causal relation leads to statistical dependence between variables, is harder to defend. A simple counterexample is the dice guessing game described above. Even though the guess of the player obviously has an influence on the outcome of the game, as demonstrated in the counterfactual example, the distribution of the variable W that indicates whether the player has won is independent of the guess of the player.

Another limitation is that the framework of Pearl (2009) does not allow for feedback loops in the data. This limitation is present in many natural systems. To illustrate this point, we use two examples. The first example is a river system for which we measure the weekly amount of water near the source and the estuary of the river. If we model this system as a causal model and consider two experiments. For the first experiment, we dump a large amount of water near the river's source. We expect to see a strong effect in both measurements in this experiment. For the second experiment, we dump the same amount of water into the river near the estuary. We expect a strong effect in only one of the measurements in this experiment. However, the increased water level near the estuary will lead to a slower flow, and, hence, this experiment will also slightly influence the amount of water upstream. In this situation, even though modeling this connection as directed in the river flow direction is only an approximation, the approximation error will be small and, hence, the model will be useful. The second example is a rope where we measure the position of both ends of the rope. To create the causal model, we again consider two interventions, one on each end of the rope. If we pull on the left end of the rope, both ends will move. If we pull on the right end of the rope, both ends will move. In this example, modeling the system with only a directional link between the position of the two ends is far from the truth and will not be useful.

Further, in Pearl (2009), the author refrains from giving a formal definition of a causal link. This lack of definition includes not not specifying between which entities we can have causal links. It is especially relevant to consider causal links between categories and their defining properties such as being an author and writing or being a researcher and conducting research. The framework of Pearl (2009) does not specify what can be used as the endogenous variables.

Another point of criticism against all frameworks that try to formalize the concept of causality without considering semantic properties of the link is that it is unclear when the notation of "causing" is transitive. An example is a causal chain that Hitchcock (2001) attributes to Hall (2004).

In this example, a huge boulder starts rolling and threatens to kill a hiker. A second hiker who notices this shouts a warning, causing the first hiker to duck. The boulder misses the hiker, which, consequently, survives.

In this story, we find the following causal chain: *Boulder starts rolling* $\rightarrow$ *the second hiker shouts* $\rightarrow$ *the first hiker ducks down* $\rightarrow$ *the boulder misses* $\rightarrow$ *the first hiker survives*. The end of this chain is transitive. Most people would agree that the hiker survived because they ducked down and that the hiker survived because the other warned him. However, nobody will claim that the hiker survived because the boulder started rolling. This discrepancy demonstrates that the concept of cause

is sometimes transitive and sometimes not, making it difficult to formalize causal relations as mathematical relations.

2.1.4 The Suitability of this Framework for the Problem Tackled in this Work

Many of these difficulties are not present in the scenario we consider in this work. In this section, we discuss which difficulties are present or absent when applying the framework of Pearl (2009) in our method. First, we use the word relevant instead of *causes* since it gives a better intuition in this exact application of the framework and to avoid confusion with the colloquial term "causes" or other definitions of causation such as the ones discussed above.

The first problem discussed above is the assumption that guarantees that if a feature is relevant for the prediction of a deep neural network, its distribution will be statistically dependent on the distribution of the prediction of the deep neural network. Above, we gave an example in which this assumption does not hold. We believe that this assumption can be violated in some deep learning problems and, hence, some features are wrongly not identified as used. However, many examples in which this happens are discrete, for example, the XOR-gate. However, these examples are uncommon in deep learning.

The second limitation named above is the feedback circles contained in natural systems. However, a supervised learning algorithm is not a natural system. In particular, it does not contain feedback. Most importantly, the prediction of the supervised learning algorithm has no influence on any feature of the data and changing it will not change the input. Hence, feedbacks are no problem in our application of the framework of Pearl (2009).

The third limitation named above is the ambiguity on what can be used as a variable in the causal model. In our method, however, this is not ambiguous. The only causal relation we investigate is the relation between features of the input and the classifier's prediction. Both of these are random variables by definition.

Further, also the problem of transitivity is not critical in our application of the causal framework because we are not trying to infer a causal relation from a chain of causal relations.

In summary, we conclude that the framework of Pearl (2009) has certain limitations. However, it is suitable for our application, namely determining whether a deep neural network uses a feature.

2.2 Machine Learning

In this section, we introduc the basic concepts of machine learning. Section 2.2.1, introduces the basic concept of statistical learning and, in particular, supervised learning and deep learning. Afterward, we explain the concept and the structure of neural networks. To this end, we introduce neural networks as a solution to the

learning problem in Section 2.2.2. More particular, we present the universal approximation theorem, which provides evidence that neural networks are especially suited for deep learning tasks.

2.2.1 The Problem of Statistical Learning

In this section, we introduce the problem of statistical learning. We follow the information from the first chapter of Vapnik (1998) and refer the reader to this work for further details.

The problem of statistical learning is choosing an optimal function from a set of functions. To this end, we need a risk functional R that maps any function f to its risk

$$\text{Risk} = R(f) \tag{2.19}$$

and a set $\mathcal{H}$ of functions from which we aim to select the one with the lowest risk. Hence, the problem is

$$\arg\min_{f \in \mathcal{H}} R(f), \tag{2.20}$$

identifying the minimum of the risk functional over the set of functions $\mathcal{H}$.

An example of this is the method of least squares for finding the best linear fit. The class of functions, in this case, is the set of linear functions $\mathcal{H} = \mathbb{R}^*$. The risk-functional is given by the mean squared error. If the inputs x and outputs y to the linear function follow the distribution $F(x,y)$, the risk-functional is given by

$$\arg\min_{f \in \mathbb{R}^*} \int (f(x) - y)^2 \, dF(x,y). \tag{2.21}$$

However, this is not yet a statistical learning problem but a problem of variational calculus. The problem of statistical learning starts if we cannot observe the distribution but have to evaluate the value of the risk functional from some samples drawn from the distribution $F(x,y)$. The risk based on this sample is called the empirical risk, and the functional evaluating it is called empirical risk functional. For the above example, it is given by

$$\text{Empirical Risk} = \sum_{i=1}^{n} R(f(x,y)) = \frac{1}{n} \sum_{i=1}^{n} (f(x_i) - y_i)^2 . \tag{2.22}$$

To this end, two problems arise. The first problem is to find the function in the set of functions that minimizes the empirical risk functional. The second problem is to select a set of functions for which the function that minimizes the empirical risk functional also minimizes the risk functional. The second question is answered in the central result of Vapnik (1998). Here we focus on the first part. More specifically, we focus on the task of supervised learning.

In statistical learning, we can differentiate between three kinds of algorithms. The three kinds are unsupervised, reinforced and supervised learning algorithms. The main difference between these kinds of learning algorithms is the supervision

signal provided to the algorithm. As described above, the goal of statistical learning is to select a function from samples of a distribution. In the first kind of learning algorithm, unsupervised learning, only the function input and not the function output are provided. Examples of these learning algorithms are clustering algorithms or density estimation. The second kind of learning algorithm, the reinforced learning algorithm, receives a quality measure for its output in addition to the inputs. A typical example of a problem solved using reinforcement algorithms is games. The algorithm has to map a game's situation onto the optimal next move. The inputs to this mapping can be observed while playing the game. However, whether the predicted move was indeed optimal cannot be observed. The learning algorithm is not provided with the correct output but with a quality measure for the suggested solution, namely, whether the game was won or not. Finally, the third kind of learning algorithm is supervised learning algorithms. For this kind of algorithm, the input and the correct output to the function are provided as a learning signal. A supervised learning algorithm can be described as a pair (T, F) of two functions. The first function, T, is called the training function. It maps the set of labeled training, examples $\{(S, Y_S)\}$, onto the weights, W, that are used to parametrize the second function F,

$$\begin{aligned} T : \mathcal{P}\left(\mathbb{S} \times \mathbb{Y}\right) &\to \mathbb{R}^m, \\ \{(S, Y_S)\} &\mapsto W. \end{aligned} \tag{2.23}$$

The second function, F, is the inference function. This function is the learned function of the statistical learning algorithm. It maps an input example I and the set of weights W onto the prediction of P

$$\begin{aligned} F : \mathbb{S} \times \mathbb{R}^m &\to \mathbb{Y} \\ (I, W) &\mapsto P. \end{aligned} \tag{2.24}$$

We illustrate these definitions using two examples. The first example is, again, a linear regression, the function T in this example is the method of least squares that maps the training examples onto the optimal coefficients. The function F multiplies the coefficients and the inputs I to find the prediction P. A second example is the k-nearest-neighbor classifier. In this example, the function T is the identity. The set of weights W is the same as the labeled training set $\{(S, Y_S)\}$. The second function, F, identifies the k-nearest-neighbors of I in W and combines their labels into a single prediction P.

In traditional shallow learning, the researcher will select and handcraft features from the inputs. These features are then used as input to, for example, a neural network. Hence, a learning algorithm consists of two distinct functions. The first function, F_f, is the function that maps an input I onto its feature representation. The second function, F_c, is the function that afterward takes in the feature representation and performs the classification or regression tasks. Hence, the approximated

function F, that maps the input onto the desired output is given by

$$F = F_c \circ F_f. \tag{2.25}$$

The main difference to deep learning, as described, for example, in Reimers and Requena-Mesa (2020), is that instead of optimizing both of these functions individually, we optimize their concatenation F directly in an end-to-end fashion. Since this requires fitting complex functions, neural networks are a suitable choice.

2.2.2 Neural Networks

The two central parts of a statistical learning problem are the risk functional and $\mathcal{H}$, the set of functions. One possibility for this set of functions and the way to parametrize and optimize these functions efficiently is deep neural networks. A deep neural network comprises multiple layers, each containing numerous neurons. A neuron is a function that maps a set of inputs $\mathbf{x}$ onto one output y. To this end, the neuron has a set of weights $\mathbf{w}$, one for every input to the neuron, and a bias variable b. Furthermore, the neuron contains an activation function σ. The neuron is given by

$$y = \sigma\left(\sum_{i=1}^{n} w_i x_i + b\right). \tag{2.26}$$

The neural network is built of multiple layers, each consisting of multiple neurons. The first layer receives the inputs to the neural network as inputs to its neurons. The subsequent layers' neurons get the outputs of the previous layers' neurons as input. Finally, the outputs of the last layers' neurons become the neural network's output.

One of the main advantages of neural networks is the universal approximation theorem that was introduced for different classes of activation functions σ. We follow the formulation of Cybenko (1989) to state the universal approximation theorem:

Theorem 1. (Universal Approximation Theorem) *Let σ be a suitable activation function. The finite sums of the form*

$$F(\boldsymbol{x}) = \sum_{j=1}^{N} \alpha_j \sigma\left(\boldsymbol{w}_j^T \boldsymbol{x} + b_j\right)$$

are dense in $\mathcal{C}(I_n)$, the continuous functions on the unit interval, with respect to the supremum norm. In other words, given any continuous function f on the unit interval I_n and any $\epsilon > 0$, there is a sum $G(\boldsymbol{x})$ of the above form, for which

$$\forall f \in \mathcal{C}(I_n) : \forall \epsilon > 0 : \forall \boldsymbol{x} \in I_n : |G(\boldsymbol{x}) - f(\boldsymbol{x})| < \epsilon.$$

These finite sums $F(\mathbf{x})$ correspond to neural networks with two layers, one hidden layer of N neurons, which uses σ as an activation function and an output layer with one neuron that uses the identity as an activation function and no bias.

Note that the n-dimensional unit interval can be replaced by any compact subset of $\mathbb{R}^n$.

This theorem is proven for multiple classes of activation functions σ. For example, Cybenko (1989) proves it for continuous sigmoidal functions in their Theorem 2 and Hornik (1991) for continuous, bounded and nonconstant functions in their Theorem 2. Furthermore, Leshno et al. (1993) prove it for every function σ, which is not an algebraic polynomial in their Theorem 1. The same was proven in Theorem 3.1 of Pinkus (1999).

Hence, a neural network can be used as a general function approximator even for complex functions with a previously unknown structure. Therefore, neural networks are suited to tackle the additional complexity resulting from combining the feature selection and the classification task in deep learning.

2.3 Test of Independence

When dealing with random variables or random processes, such as the training of a neural network using stochastic gradient descent or the endogenous variables of a structural causal model, it is not trivial to determine whether an observed difference is due to an effect or whether the differences are just due to the internal variance of the variables.

Statistical tests determine whether an effect is due to randomness or whether it is significant. In a statistical test, we are presented with two hypotheses. The first hypothesis, H_0, is called the null hypothesis. The null hypothesis is the hypothesis we will assume is true unless we find it highly unlikely given the data. In that case, we assume the alternative hypothesis H_1. To make this decision, we need to decide the probability at which we decide that H_0 is highly unlikely. This number is called the level of significance. The most common choice for the level of significance is 0.05, due to a subjective choice of R. A. Fisher, who wrote:

> "The value for which $P = .05$, or 1 in 20, is 1.96 or nearly 2; it is convenient to take this point as a limit in judging whether a deviation is to be considered significant or not. Deviations exceeding twice the standard deviation are thus formally regarded as significant. Using this criterion we should be led to follow up a negative result only once in 22 trials, even if the statistics are the only guide available. Small effects would still escape notice if the data were insufficiently numerous to bring them out, but no lowering of the standard of significance would meet this difficulty." (Fisher, 1925, p. 47)

Nevertheless, the level of significance has to be decided for every situation individually.

In this work, we use statistical dependence tests to decide whether two distributions are dependent. For these tests, the null hypothesis H_0 is that the two distributions are independent, while hypothesis H_1 is that the two distributions

are dependent. We estimate the likelihood of both hypotheses given the data. To this end, we assume a prior probability of 0.5 for each hypothesis. Since the observed data is fixed, we can use Bayes' theorem to prove that the likelihood of the hypothesis given the data is proportional to the likelihood of the data under the hypothesis.

Because finding the likelihood for the data is often untraceable, we reduce it to the likelihood of one feature of the data. Ideally, this feature is scalar and has a different distribution under the different assumptions. Further, it usually is larger under assumption H_1 than for assumption H_0, such that we can evaluate the later tests only to one side. This feature is called the test statistic, and selecting it is the main difference between different dependence tests.

To test the dependence of two random variables, we need to evaluate the test statistic distribution under the assumption of each hypothesis. This evaluation is especially hard for H_1, the assumption of dependence. Since we do not specify the nature of the dependence, we need to approximate the distribution for each possible dependence simultaneously. To this end, we assume a uniform distribution of the test statistic under the assumption H_1 meaning any data has the same likelihood. We use a shuffle test to evaluate the distribution under the null hypothesis H_0. If the two variables are independent, the probability of the observed data, $(X_i, Y_i)_{i\in\mathcal{I}}$, equals the probability of $(X_i, Y_{\pi(i)})_{i\in\mathcal{I}}$ for any permutation π. Hence, we can find the distribution of the test statistic under H_0, the assumption of independence by calculating the test statistics for all possible permutations π, or approximate the test statistic distribution by calculating it for many permutations.

Since every set of observations has the same likelihood under the assumption H_1, we want to evaluate the likelihood under hypothesis H_0. To this end, we assess the probability of observing data with an equal or higher test statistic in the distribution under the assumption H_0 that we evaluated as described above. This value is called the p-value.

Note that if assumption H_0 is true, the p-value will be uniformly distributed on the interval $[0, 1]$, while if hypothesis H_1 is true, the p-value will be very low. As described above, we will reject the null hypothesis H_0 if the p-value is lower than our selected level of significance.

Two things are important when using and interpreting the results of the method of statistical dependence test. First, since the p-values are uniformly distributed if the null hypothesis is true, we will always encounter false positives, where we reject the null hypothesis even though it is true. The fraction of these false positives is equal to our level of significance. Second, since the p-value is uniformly distributed if the null hypothesis is correct, it does not make sense to compare the p-value of tests for different variables and argue that one is "more dependent" because its test resulted in a lower p-value. A statistical dependence test only returns a binary result, and the p-value should never be interpreted further than this binary decision.

2.3.1 Correlation and Partial Correlation

The first test statistic we consider is the correlation between the two variables. The correlation is given by

$$\rho_{X,Y} = \frac{\mathbb{E}(XY) - \mathbb{E}(X)\mathbb{E}(Y)}{\sqrt{\mathbb{E}(X^2) - \mathbb{E}(X)^2}\sqrt{\mathbb{E}(Y^2) - \mathbb{E}(Y)^2}}. \tag{2.27}$$

While the correlation for independent variables is zero, the correlation for dependent variables is not necessarily bigger than zero. The correlation captures only linear relations between the variables. Hence, it is an independent test only for certain classes of distributions. For example, if all involved variables are Gaussian. The main advantage of correlation as a dependence test is that it is fast to calculate, even for high dimensional variables. Further, it is interpretable in the form of the coefficient of determination. To turn the correlation into a conditional dependence test, we use the partial correlation instead. The partial correlation between the two variables X and Y given Z is calculated by first finding the best linear fit from Z onto X,

$$\alpha_X = \arg\max_{\alpha}(\alpha(Z - \mathbb{E}(Z)) + \mathbb{E}(X) - X)^2, \tag{2.28}$$

and the best fit from Z onto Y,

$$\alpha_Y = \arg\max_{\alpha}(\alpha(Z - \mathbb{E}(Z)) + \mathbb{E}(X) - X)^2. \tag{2.29}$$

Afterward, we calculate the correlation between the residuals of X and Y, meaning the part of X and Y that can not be explained by Z. Formally, the residuals are defined as

$$\hat{X} = X - (\alpha_X(Z - \mathbb{E}(Z)) + \mathbb{E}(X)) \tag{2.30}$$

and

$$\hat{Y} = Y - (\alpha_Y(Z - \mathbb{E}(Z)) + \mathbb{E}(Y)). \tag{2.31}$$

Then the partial correlation is given by

$$\rho_{\hat{X},\hat{Y}\cdot Z} = \rho_{\hat{X},\hat{Y}} = \frac{\mathbb{E}(\hat{X}\hat{Y}) - \mathbb{E}(\hat{X})\mathbb{E}(\hat{Y})}{\sqrt{\mathbb{E}(\hat{X}^2) - \mathbb{E}(\hat{X})^2} - \sqrt{\mathbb{E}(\hat{Y}^2) - \mathbb{E}(\hat{Y})^2}}. \tag{2.32}$$

The test statistic is calculated as the coefficient of determination $\rho_{\hat{X},\hat{Y}}$ between $\hat{Y}$ and $\hat{X}$. As described above, we perform a shuffle test to check whether the correlation is significant. Since the partial correlation is based on the correlation, it shares the same advantages and disadvantages. First, it is not a mathematical independence test in general. It only captures linear connections between the variables. However, as shown, for example, in Baba et al. (2004), this includes relevant cases such as the case where all distributions are multivariate Gaussian. Further, the partial correlation can be calculated fast, even for high dimensional variables, similar to the correlation. Finally, the test statistic of the partial correlation

can be interpreted. Its square denotes the fraction that is explained by X of the variance in Y that can not be explained by Z.

2.3.2 Maximum Correlation Coefficient

As described above, in general, the correlation and partial correlation are no statistical dependence or conditional statistical dependence test, respectively. One possibility to extend the idea of correlation to a general statistical dependence test is the maximum correlation coefficient introduced by Sarmanov (1958). To calculate the maximum correlation coefficient, we transform both random variables using an arbitrary function to maximize their correlation. Formally, the maximum correlation coefficient is given by

$$\begin{aligned} \text{MCC}(X,Y) &= \sup_{f,g} \rho_{f(X),g(Y)} \\ &= \sup_{f,g} \frac{\mathbb{E}(f(X)g(Y)) - \mathbb{E}(f(X))\mathbb{E}(g(Y))}{\sqrt{\mathbb{E}(f(X)^2) - \mathbb{E}(f(X))^2}\sqrt{\mathbb{E}(g(Y)^2) - \mathbb{E}(g(Y))^2}}. \end{aligned} \tag{2.33}$$

The maximum correlation coefficient is zero if and only if the two variables are independent. Hence, the maximum correlation coefficient is a general statistical dependence test. However, calculating it requires fitting two arbitrary functions, which is a challenging problem by itself.

To transform the maximum correlation coefficient into a conditional dependence test. We substitute the correlation for a partial correlation

$$\sup_{f,g} \rho_{f(X),g(Y)\cdot Z}. \tag{2.34}$$

As described above, to calculate the partial correlation, we use the best linear fits $f_X(Z)$ and $f_Y(Z)$. Calculating a linear fit is not sufficient to create a general conditional dependence test. Instead, we use an additional function h which transforms Z to get

$$\text{MPCC}(X,Y \mid Z) = \rho_{f(X),g(Y)\cdot h(Z)}. \tag{2.35}$$

This conditional dependence test has the same drawbacks as the unconditional version of the maximum correlation coefficient. However, the high complexity of fitting functions is an even more severe drawback because one more function has to be fitted. For this reason, the maximum correlation coefficient is suitable in situations where we have some prior information on the structure of the possible functional connections between variables.

Another test that is based on fitting functions between the involved variables is the Fast Conditional Independence Test (FCIT) proposed by Chalupka et al. (2018). However, in contrast to the maximum correlation coefficient, we only have to fit two functions in the FCIT. The reason is that FCIT builds on Doob's conditional independence property (Kallenberg, 1997, Proposition 5.6)

$$X \perp\!\!\!\perp Y \mid Z \Leftrightarrow \mathbb{P}(Y \mid X,Z) = \mathbb{P}(Y \mid Z) \quad a.s. \tag{2.36}$$

Consequently, this test uses two decision trees, one to predict the value of Y from only Z and the other predicts the value of Y from X and Z. The test statistic is then given by

$$\text{Var}(Y - \mathbb{E}(Y \mid Z) - \text{Var}(Y - \mathbb{E}(Y \mid X, Z)). \tag{2.37}$$

The advantage of this procedure is that we have to fit only two functions. However, FCIT is no statistical dependence test. An obvious counterexample, where a dependence goes undetected, is given by $Y \sim \mathbb{U}[-1, 1]$, $X = Y^2$ and $Z = 0$. In this case $\mathbb{E}(Y \mid Z) = \mathbb{E}(Y \mid X, Z) = 0$ and hence the test statistic is zero. However, X and Y are clearly dependent. However, it can correctly detect many dependencies and is very fast compared to the other tests described in this section. Therefore, it can be used in situations where many tests need to be performed.

2.3.3 Mutual Information and Conditional Mutual Information

The next quantity we can use as a test statistic is the mutual information between the two variables. For two independent variables, X and Y, the joint distribution factorizes into

$$\mathbb{P}(X, Y) = \mathbb{P}(X)\mathbb{P}(Y). \tag{2.38}$$

As the test statistic, we, hence, use the Kulback-Leibler divergence between the joint distribution of X and Y and the product of their marginal distributions

$$\text{MI}(X, Y) = D_{KL}(\mathbb{P}(X, Y); \mathbb{P}(X)\mathbb{P}(Y)) = \int_{\mathcal{X}} \int_{\mathcal{Y}} p_{X,Y}(x, y) \log \left(\frac{p_{X,Y}(x, y)}{p_X(x) p_Y(y)} \right) dxdy. \tag{2.39}$$

As described above, for independent variables, the two distributions in the Kulback-Leibler divergence are the same, and, hence, the test statistic is zero. However, if the two variables are not independent, the mutual information will be positive as shown by (Wyner, 1978, Lemma 3.1).

The main drawback of using the mutual information as a dependence test is that we must approximate the joint distribution and the two marginal distributions from the data. The fraction of the distributions is the focus of these approximations. To this end, it is essential to approximate small values accurately. To approximate distributions, in particular distributions with a different number of dimensions consistently, is a complex problem. Hence, this independence test is best suited if we have prior knowledge of the distributions, which allows us to approximate the distributions efficiently.

The conditional version of the mutual information is the conditional mutual information. The formula of the conditional mutual information is the same as for the mutual information, but for the joint distribution, we also include the variable Z on which we condition. Resulting in a joint distribution of three variables. In the marginal distributions, we still marginalize over one of the variables and, therefore, get the joint distribution of each variable together with the variable we condition

on. Consequently, the conditional mutual information is given by

$$\begin{aligned} \mathrm{CMI}(X, Y \mid Z) &= D_{KL}(\mathbb{P}(X, Y, Z); \mathbb{P}(X, Z)\mathbb{P}(Y, Z)) \\ &= \int_{\mathcal{X}} \int_{\mathcal{Y}} \int_{\mathcal{Z}} p_{X,Y,Z}(x, y, z) \log \left(\frac{p_{X,Y,Z}(x, y, z) p_Z(z)}{p_{X,Z}(x, z) p_{Y,Z}(y, z)} \right) dxdydz. \end{aligned} \tag{2.40}$$

The conditional mutual information intensifies the drawbacks of unconditional mutual information. The weakness that we have to approximate distributions of different dimensionality consistently is still present, but the dimensionality of each distribution is higher than in the unconditional case. Hence, especially for the conditional case, this dependence test is suitable if we have prior information on the form of the distribution, for example, if we know that they are from a parametrized family of distributions.

2.3.4 Kernel Independence Tests

One of the central principles in data science was formulated by Vapnik (1998):

> "If you possess a restricted amount of information for solving some problem, try to solve the problem directly and never solve a more general problem as an intermediate step. It is possible that the available information is sufficient for a direct solution but is insufficient for solving a more general intermediate problem." (Vapnik, 1998, page 12)

However, the dependence tests we introduced before violate this principle. In the maximum correlation coefficient described in Section 2.3.2, we do not only calculate the test statistic, but as an intermediate step, we approximate two, or in the conditional case three, functions that connect the variables. Note that the information on the functions is enough to calculate the test statistic, but knowing the test statistic is not enough information to infer the three functions. Hence, the problem of approximating the functions is more general than the problem we are trying to solve. In the cases of mutual and conditional mutual information introduced in Section 2.3.3, we have to approximate the joint, and the marginal distribution of multiple variables. Similar to the above case, the test statistic can easily be calculated from the distributions. However, knowing the test statistic is not enough information to infer all distributions. Consequently, the problem of approximating the distributions can be considered a more general problem than estimating the mutual information.

This section, follows the principle more closely and evaluates the test statistic directly. The statistical independence test that follows this idea is the Hilbert-Schmidt independence criterion.

The idea of this independent test is to evaluate whether, for the pairs (X_i, Y_i), the value of the first variable X_i is similar to the first value X_j of the same pairs of which the second value Y_j is similar to Y_i. To this end, we start by calculating the kernel matrix for each variable, K_X for the variable X and K_Y for the variable Y. By definition, each entry of the kernel matrix $k_{ij} \in K_X$ contains the similarity between

the values X_i and X_j. To make the kernel matrices comparable, we normalize them. We do this by multiplying the kernel matrix with a normalizing matrix H, whose entries are given by

$$h_{ij} = \delta_{ij} - m^{-2} \tag{2.41}$$

with δ denoting the Kronecker-delta and m is the number of data points. To calculate the test statistic, we, afterward, sum up the dot products between the similarity vectors corresponding to each pair (X_i, Y_i), namely the i-th collum of the two normalized kernel matirices

$$\sum_{i=1}^{m} \langle (K_X H)_i, (K_Y H)_i \rangle. \tag{2.42}$$

Note that the dot product of normalized vectors is related to the cosine similarity between the vectors. If the variables are dependent, the similarity structure in both spaces is the same. Hence, this value will be high for dependent variables but small if the variables are independent. Since the kernel matrices are symmetric, using the i-th column of the kernel matrix is the same as using the i-th row. Further, the dot product of the i-th collum of the kernel matrix K_X and the i-th row of the kernel matrix K_Y is the i-th element of the diagonal of their product. Summing them up is equivalent to calculating the trace of this matrix

$$\sum_{i=1}^{m} \langle (K_X H)_i, (K_Y H)_i \rangle = \mathrm{Tr}\left(K_X H K_Y H\right). \tag{2.43}$$

We multiply this value with a normalizing factor to calculate the test statistic

$$\mathrm{HSIC}(X, Y) = \frac{1}{(m-1)^2} \mathrm{Tr}\left(K_X H K_Y H\right). \tag{2.44}$$

The Hilbert-Schmidt independence criterion was introduced by Gretton et al. (2007). However, they use a different way to derive it and prove that it is a statistical dependence test, meaning that the test statistic is zero if and only if the variables are independent. To this end, the authors of that paper relate the formula presented above to the Hilbert-Schmidt norm of the cross-covariance operator between the two kernel spaces. When the kernel spaces of both variables are universal reproducing kernel Hilbert spaces, meaning that the functions in the kernel spaces are dense in the space of bounded continuous functions, the largest singular value of the cross-covariance operator, $||C_{XY}||$, is equal to zero if and only if $X \perp\!\!\!\perp Y$.

As described above, the great advantage of the dependence test based on the Hilbert-Schmidt independence criterion is that we do not approximate any more information than the test statistic. However, to calculate it, we have to select a suitable kernel, and we have to calculate and multiply the kernel matrices. The former is difficult. The similarity decoded in the kernel has to be sensitive enough to capture the relations between the variable but not such sensitive that it picks up finite data effects. The higher the sensitivity of a kernel, for example, a universal

kernel such as the radial basis functions kernel, the more data we need to distinguish between genuine and spurious relations. Unfortunately, the second drawback prohibits us from using this dependence test in large datasets. If we have many data points, creating the kernel matrix will scale quadratically in the number of inputs, $\mathcal{O}(m^2)$, and multiplying these matrices will scale like $\mathcal{O}(m^{2.3728596})$ (Alman and Williams, 2021). Hence, for large datasets, evaluating this criterium becomes infeasible.

To turn this test into a conditional dependence test, Fukumizu et al. (2007) replace the cross-covariance similar to extending the correlation to the partial correlation in Section 2.3.1. To this end, instead of evaluating the similarity between the kernel matrices K_X and K_Y directly, we first remove the similarity that can also be observed in K_Z, the kernel matrix of the variable we condition on. This calculation leads to the formula

$$\text{HSICONIC}(X, Y \mid Z) = \text{Tr}\left(\left(HK_XH - HK_XHHK_ZH\right)\left(HK_YH - HK_YHHK_ZH\right)\right) \tag{2.45}$$

for the Hilbert-Schmidt conditional independence criterion. This straightforward calculation, however, is numerically unstable. Hence, different authors have suggested different approximations of this formula, that are more stable to evaluate. For example, the approximation

$$(1 - x) \approx \frac{1}{1 + x} \tag{2.46}$$

that holds for small values of x leaves us with

$$\text{HSCONIC}(X, Y \mid Z) \approx \text{tr}\, G_X S_Z G_Y S_Z. \tag{2.47}$$

Here, for $A \in \{X, Y, Z\}$, we use $G_A = HK_AH$ and $S_Z = (\mathbb{I} + 1/mG_Z)^{=1}$ with the identity matrix $\mathbb{I}$.

As for the other dependence tests described here, the drawbacks of the unconditional dependence tests are intensified in the conditional version. Specifically, we have to select three suitable kernels, and since we have to perform more matrix multiplications, the bad scaling in the number of samples becomes a more severe problem. Further, the conditional version of this test has some numerical stability issues. Hence, numerically more stable approximations to this criterion are used in practice. For more information, we refer the reader to Fukumizu et al. (2007).

Two additional dependence tests that build on this idea are the Randomized Conditional Independence Test (RCIT) and the Randomized conditional Correlation Test (RCoT) proposed by Strobl et al. (2019). They substitute the radial basis function, which can be understood as an infinite sum of Fourier features, by an approximation with a finite selection of Fourier features. These features are selected randomly. This approximation allows for better scaling in the number of examples and hence, for this test to be used even for large datasets. However, we encounter another error rising from the approximation. Fortunately, Strobl et al. (2019) demonstrated that this error is small. For further information, we refer the

reader to their work.

2.3.5 Predictability

A common idea in determining whether variables are dependent is to check whether the value of one can be predicted from the other. For two variables B, R, we can quantify this, for example, by the mean squared error

$$\min_f \int (b - f(r))^2 \, dp_{B,R}(b,r) \tag{2.48}$$

where the function f is typically parameterized, for example, by a deep neural network. Predictability is not an independence criterion. However, the predictability criterion is similar to the maximum correlation criterion. Since predictability is not an independence criterion, we can not turn it into a conditional dependence criterion. However, we can extend the predictability criterion, similar to the maximum correlation coefficient to the conditional case by replacing the correlation with the partial correlation.

We start by showing the connection between the predictability criterion and the maximum correlation coefficient. To this end, we first consider the case where the function f is linear and both variables are univariate,

$$\min_f \int (b - f(r))^2 \, dp_{B,R}(b,r) = \min_{\alpha,\beta} \int (b - \alpha r - \beta)^2 \, dp_{B,R}(b,r). \tag{2.49}$$

Since the optimal parameter for β will center both variables in this case, this is the same as the variance between the bias variable B and the best linear prediction of B given the prediction R,

$$\min_{\alpha,\beta} \int (b - \alpha r - \beta)^2 \, dp_{B,R}(b,r) = \min_{\alpha} \mathrm{Var}\left(B - \alpha R\right). \tag{2.50}$$

We use this functional as a loss. The relevant properties for this use are the minima and the relative values of the functions. None of these properties change if we scale the function by a constant, positive factor. Since the choice of the function f has no influence on the variable B, the variance of B is such a constant, positive factor. Hence, we can simply scale the function by this factor and get

$$\min_{\alpha} \frac{\mathrm{Var}\left(B - \alpha R\right)}{\mathrm{Var}\left(B\right)}. \tag{2.51}$$

Further, instead of minimizing, we can maximize the negation, leading to

$$\max_{\alpha} 1 - \frac{\mathrm{Var}\left(B - \alpha r\right)}{\mathrm{Var}\left(B\right)}. \tag{2.52}$$

This is the definition of the coefficient of determination (Egert, 2012), which, in this

linear case, is the square of the Pearson correlation coefficient,

$$\max_{\alpha} 1 - \frac{\text{Var}\,(B - \alpha R)}{\text{Var}\,(B)} = \rho^2_{B,R} = \text{Corr}\,(B, R)^2\,. \tag{2.53}$$

To generalize this relation from the linear to the non-linear case, we find the functions f that maximizes this correlation

$$\max_{f} \text{Corr}\,(B, f(R))^2\,. \tag{2.54}$$

This loss function was, for example, used for adversarial debiasing by Adeli et al. (2021). However, this is not a dependence test, more specifically, this can be zero even if the two variables are independent. For example, if the prediction R is uniformly distributed on the interval $[-1, 1]$

$$R \sim \mathcal{U}[-1, 1] \tag{2.55}$$

and the bias variable B is given by

$$B = R^2. \tag{2.56}$$

Obviously, the two variables are not independent, but for any function f, the covariance between B and R is given by

$$\text{Cov}(B, f(R)) = \int b f(r) dp_{B,R}(b, r). \tag{2.57}$$

Substituting B by definition (2.56), we get

$$\int b f(r) dp_{B,R}(b, r) = \int_{-1}^{1} r f(r^2) dr. \tag{2.58}$$

Splitting the interval at zero, changing the variable in the first half from r to $-r$, exchanging the borders of the integral and reuniting the two integrals evaluate this integral to

$$\int_{-1}^{1} r f(r^2) dr \tag{2.59}$$

$$= \int_{-1}^{0} r f(r^2) dr + \int_{0}^{1} r f(r^2) dr \tag{2.60}$$

$$= \int_{0}^{1} -r f((-r)^2) dr + \int_{0}^{1} r f(r^2) dr \tag{2.61}$$

$$= \int_{0}^{1} (r - r) f(r^2) dr \tag{2.62}$$

$$= 0. \tag{2.63}$$

Hence, the loss functional $\max_f \text{Corr}(B, f(R))^2$ is equal to zero, although the

two variables are not independent. This example demonstrates that not being useful for prediction is not a criterion for independence.

This criterion is related to the maximum correlation coefficient described in Section 2.3.2. This criterion uses the following maximum as a test statistic,

$$\mathrm{MCC}(R, B) = \max_{f,g} \mathrm{Corr}(f(R), g(B)). \tag{2.64}$$

To turn this into a conditional independence test, we use the partial correlation

$$\mathrm{MPCC}(R, B \mid L) = \min_{h} \max_{f,g} \rho_{f(R),g(B) \cdot h(L)}. \tag{2.65}$$

2.4 Attribution and Visualization

In the shallow learning setting, a researcher selects the features that a learning algorithm uses to perform its classification. One advantage of this approach is that it is easy to understand which feature the algorithm uses to reach its prediction. However, as described, for example, in Reimers and Requena-Mesa (2020), in deep learning, selecting features and classifying are combined into one process. This combination makes it difficult to understand, whether a feature is used by the algorithm to make its prediction.

Researchers have developed many methods to extend this advantage of shallow approaches to the deep learning approaches. The most common method is creating a saliency map. In a saliency map, we assign a salience value to each input quantifying its importance. Note that the individual inputs are often not meaningful features in the deep learning setting, but the meaningful features are aggregation functions of multiple inputs. For example, in an image, we do not expect an individual pixel to be important, but a pattern of pixels, representing some higher level feature. Multiple different approaches on how to quantify the salience of inputs have been proposed in the literature. We introduce and compare them in Section 2.4.1. Afterward, in Section 2.4.2, we present alternatives to saliency maps proposed to determine whether a feature is relevant in the classifier's decision.

2.4.1 Saliency-Based Methods

The main idea of saliency maps is to create a saliency or importance value for each input. In the setting of images, this means creating a saliency value for each pixel. These saliency values can then be arranged as the original image to highlight important areas in the image. However, defining how important a pixel is is difficult. At the moment, we do neither have a way to evaluate it empirically nor is there widespread agreement on the theoretical definitions. Considering these difficulties, multiple ways to derive saliencies are proposed, leading to different results. The three main ways are using the gradient of the deep neural network, which we discuss

in Section 2.4.1.1, measuring the sensitivity of the output to replacing some of the inputs we consider in Section 2.4.1.2 and methods that approximate the effect of perturbations using a first-order Taylor approximation. We describe these methods in Section 2.4.1.3. Finally, we summarize the drawbacks of saliency maps in general in Section 2.4.1.4.

2.4.1.1 Gradient Based Methods

The first idea that can be used to assign a salience value to each pixel is to use the gradient. To motivate the use of the gradient as saliency, we use the example of a linear regressor. In a linear regressor, the gradient of the output given one input variable is the linear coefficient, and, hence, the gradient explains how much the output changes if the input changes by one unit.

Multiple methods were proposed that use the gradient or slight variations of the gradient of the neural network's output depending on each input as the salience of that input. For example, Zeiler and Fergus (2014) use the derivative for all parts of the neural network except for the ReLU non-linearities, for which they use the ReLU again instead of its derivative.

This method of forming saliency maps has some drawbacks. The most severe drawback is that the derivative, by definition, is highly local. However, suppose we want to understand whether a feature is important. In that case, we are not just interested in whether a minimal increase in the feature will lead to a minimal change in the prediction. To demonstrate this drawback, we use the following example. This example looks at the relation between two features, X and Y, and the probability P of the example being classified as class C. The relation between the features and the outputs is given by

$$P = \frac{\sin(1000X)}{100} + \frac{0.99}{1 + \mathrm{e}^{1000Y}}. \tag{2.66}$$

In this situation, the derivative of P for X can be as high as 10, but the feature is meaningless towards the algorithm's decision, as the influence is at most 0.01. In contrast, for any value of Y with $|X| > 0.01$, the derivative of P for Y is smaller than 10^{-3}. Nevertheless, the feature is obviously very relevant to the classifier's decision. Hence, we find that using the gradient or derivative of a function can lead to local solutions that might not reflect the global behavior of the function enough to be used as a saliency.

In addition to these specific drawbacks of using the derivative to quantify the saliency, methods following this idea have more general drawbacks, which they share with all saliency map methods. We describe these drawbacks in Section 2.4.1.4.

2.4.1.2 Obfuscation Based Methods

The second common way to derive salience values is to obfuscate the image. The main idea is to delete some of the inputs and replace them with something mean-

ingless. To find the saliency of the replaced inputs, we take the difference between the prediction of the original and the manipulated sample.

The main difference between the various methods that employ obfuscation to create saliency maps for deep neural networks is the replacement for the obfuscated inputs. The suggestions start at straightforward ideas such as replacing areas in an image with a black or gray box as suggested in Zeiler and Fergus (2014), or noise as suggested, for example, in Dabkowski and Gal (2017) and end with more complicated ideas. For example, Zintgraf et al. (2017) replace the patches from the inputs by patches from the training set, which have similar neighboring pixels, by sampling from the knockoff distribution (Barber and Candès, 2015) as suggested by, for example, Popescu et al. (2021) or using a generative adversarial network (Goodfellow et al., 2014a) for gap-filling as suggested, for example, in Agarwal and Nguyen (2020).

This approach to creating saliency maps has some drawbacks. The first drawback is that the result of the approach depends on how we replace the obfuscated pixels. If we, for example, replace parts of an image with black boxes, light areas will receive a higher saliency than dark areas. The second problem is that this method can only detect localized features within the image. Meaning that we can use it to detect the meaningfulness of, for example, an eye pattern to classify an image, but not, for example, a distributed pattern such as the symmetry between two eye patterns. Third, obfuscation is, in many situations, too global.

To demonstrate this global scale, we use an example similar to the previous one. In the example, the probability P is given by

$$P = \frac{99\sin(1000X)}{100} + \frac{0.01}{1+\mathrm{e}^{1000Y}}. \tag{2.67}$$

Here, the more important feature is obviously X since the effect of Y is limited to 0.01. However, if we compare the values for high and low values of Y, there is no difference. In contrast, the probabilities P for high and low values X are significantly different.

In addition to these drawbacks, which are specific to creating a saliency map using obfuscation, these methods also have some drawbacks shared by all methods that create saliency maps. We discuss these drawbacks in Section 2.4.1.4.

2.4.1.3 Methods based on Taylor-extensions

As described in the two previous sections, using the gradient to produce a saliency map is, in many situations, too local, while using the distribution and replacing inputs by other values from that distribution is, in some situations, too global. Hence, in this section, we present a compromise that uses the values, and the gradient to create the salience map. The main idea is to look at the function that maps a perturbation of the input onto the output change. Since this function is unknown, we approximate it by a first-order Taylor approximation

$$\Delta P \approx F'(x_0)(x_0 - x). \tag{2.68}$$

To create the saliency map, we evaluate this difference function at zero. This evaluation is because we expect zero to be a good approximation of a point at which each class has the same probability.

This compromise between the local and global approaches to creating saliency maps is the most common approach. It was suggested, for example, by Bach et al. (2015) and in Lapuschkin et al. (2019). Further, it was adapted to the special case of deep convolutional neural networks by, for example, Mopuri et al. (2018) and Selvaraju et al. (2016). Further, for example, Montavon et al. (2017) investigate whether the results improve if zero is replaced by a close-by point, at which all classes have the same probability. However, they find that the additional difficulties and ambiguities counteract the benefits of this approach. Saliency maps created using this approach share the general drawbacks of saliency maps we describe in Section 2.4.1.4.

2.4.1.4 Drawbacks of Saliency Maps

All methods that use saliency maps to determine which features are used by a deep neural network share some common drawbacks. It is very difficult to evaluate methods that determine which feature deep neural networks use to find their prediction. Hence, we rely on other properties such as consistency to evaluate whether these methods work correctly. To this end, we check whether saliency maps lead to explanations that have decent properties. First of all, saliency maps cannot explain and are not consistent with the phenomenon of adversarial examples, which we introduce in Section 2.5. Second, it is shown by Adebayo et al. (2018) that the saliency maps created for neural networks with random weights are similar to the saliency maps created for neural networks with trained weights. Third, the reaction of the saliency map to constant shifts is not consistent, as is shown by Kindermans et al. (2019). Fourth, Ghorbani et al. (2019) show that, saliency maps are also vulnerable to adversarial examples, similar to deep neural networks. The most important drawback, however, is that saliency maps can only be used for features that are represented by a region of the image. An example of such a feature would be an eye pattern. In contrast, saliency maps can not be used to determine whether a feature such as symmetry, that is not connected to one specific, but rather to the relation of multiple regions, is used by the deep neural network.

2.4.2 Other Methods

While saliency map-based methods, which we described in the previous section, are the most common method to interpret deep neural networks, other methods exist. In this section, we introduce some alternatives.

2.4.2.1 Quantitative Testing with Concept Activation Vectors (TCAV)

The first alternative method we describe here is proposed by Kim et al. (2018). Saliency maps are a local explanation method in a diffeent sense: They only explain

the behavior of the classifier for the exact input on which it was used. In contrast, TACV is a method that aims at a global explanation for the classifier, meaning that it explains the behavior of a classifier as a whole. More specifically, TCAV aims to determine whether a specific, predefined concept C is used by a deep neural network F in its prediction. A concept is any feature that partitions all images into two parts: images that contain the concept and images that do not contain the concept. As such, a concept admits a natural binary classification task. For a given deep neural network F, one of the output classes of this deep neural network and a concept, the question that is answered with TACV is whether the deep neural network uses the concept to recognize the class. To this end, the activations in some intermediate layer are used as a representation R of the inputs. On these representations, we train a linear classifier to distinguish between examples that contain the concept and examples that do not contain the concept. The unit vector v_C^R that is orthogonal to this decision boundary is called the concept activation vector. The importance of the concept is then evaluated as the directional derivative of the detector of the class in the direction of the class activation vector.

More formally, we can understand F as a concatenation of two functions $F = F_2 \circ F_1$. The first function F_1 is the feature extractor

$$F_1 : \mathbb{R}^n \to \mathbb{R}^m \tag{2.69}$$

$$I \mapsto R \tag{2.70}$$

and the function F_2 is the classifier

$$F_2 : \mathbb{R}^m \to \mathbb{R}^c \tag{2.71}$$

$$R \mapsto P. \tag{2.72}$$

Then the importance S of a concept towards the classification of the class is given by the directional derivative of the classifier F_2 in the direction of the class activation vector

$$S = \mathrm{D}_{v_C^R} F_2 = \left\langle \nabla F_2, v_C^R \right\rangle . \tag{2.73}$$

2.4.2.2 Explaining Classifiers with Causal Concept Effect (CaCE)

Another idea to understand classifiers, meaning to understand whether the classifier extracts a feature from the data and uses it as input for its classification, is causal concept effect (CaCE). This idea is introduced by Goyal et al. (2019). Their idea is to build a generative model, for example, the decoder of an autoencoder or a GAN (Goodfellow et al., 2014a) to create an input image that is the same in every aspect but differs with respect to containing or not containing the concept in question. The prediction for the original input and the prediction for the altered input are compared, and the difference is considered the causal concept effect. For important concepts, we expect this effect to be high.

The main drawback of this method is the dependence on a generative model. First, in comparison to predictive models, which try to approximate a set of condi-

tional probabilities, generative models have to capture the whole joint probability. Hence, the problem of training a generative model is more difficult than the problem of training a predictive model. Second, if different features are correlated in the dataset, then the generative model will not be able to differentiate between them.

2.4.2.3 Methods that Link Intermediate Representations to Concepts

While it is difficult to understand the decision process of a deep neural network in a semantically meaningful way, it is easy to understand how the decision is calculated from the intermediate representations in a mathematical way. Hence, one way of solving the task of understanding which feature is used by the deep neural network is to link the intermediate representations to semantic concepts. For example, Narendra et al. (2018) employ causal methods to a deep neural network. They treat every node as a binary encoding for a concept. The same approach is discussed in Harradon et al. (2018). However, they find that the connection between single neurons and semantic concepts is unclear. More specifically, some neurons can not be connected to semantic concepts but specific pixel patterns. Furthermore, multiple neurons might represent the same concept, and linear combinations of multiple neurons can represent some concepts. Another approach in this direction is presented in Stomberg et al. (2021). In this approach, an intermediate representation is clustered, and then each cluster is linked to a semantic concept.

2.4.2.4 Feature Visualization

Another method developed to understand deep neural networks is the feature visualization framework presented by, for example, Erhan et al. (2009), Simonyan et al. (2013), Mordvintsev et al. (2015), Olah et al. (2017). The goal of feature visualization is twofold. On the one hand, it aims to get prototypical images for the classes the deep neural network can identify or, in other words, to understand how the deep neural network "expects" a prototypical example of the class to look like. On the other hand, it helps to understand what information is detected by individual neurons.

While methods of this kind achieve impressive and interesting images, even the basic assumptions underlying this method as an interpretation tool for deep neural networks are problematic. First, a classifier learns to discriminate between classes. As such it does not have to learn prototypes for the classes it can identify. Hence it is unclear why a prototypical image for every class should be extractable. Second, the notion of a prototypical image indicates that the approximated function is concave, meaning the score the deep neural network assigns to a class is at least close to monotone in the distance between the input image and the prototype. However, this is not the case for deep neural networks in general. Third, it is not clear why every neuron in the deep neural network should correspond to a semantic concept. This is a special case of the understanding, that subsequent layers of deep neural networks extract semantic features of increasing complexity. This idea is, however not compatible with the concept of adversarial examples as described

in Section 3. Fourth, many concepts are not explained by single neurons but by linear combinations of neurons (Szegedy et al., 2013), and a single neuron can be activated by images containing vastly different concepts (Olah et al., 2017). Finally, an unrestricted optimization of the input image to maximize the output of a neuron does not converge to a meaningful image but seemingly random noise patterns. Strong regularizations are necessary to arrive at images from which concepts can be interpreted. To this end, authors have employed heavy L_2-regularization, for example, Simonyan et al. (2013), a mix of different regularization methods as, for example, Olah et al. (2017), or even more complex image statistics, for example, Mordvintsev et al. (2015). The selection of the regularization has a large influence on the result and, hence, a large part of the explanations created by these methods are determined by the implementation of the explanation method rather than the deep neural network it aims to explain.

2.4.2.5 Explanation by Example

A class of models that are intuitively explainable are nearest neighbor classifiers. These classifiers simply compare the input at test time to all training inputs, select the k closest inputs and predict the most common label among these for the test input. To explain of this classifier's decision, one can look at the closest examples from the training data.

The core of these methods is the definition of similarity. Obviously, when classifying, for example, natural objects in images, the Euclidean distance is not a good measure for similarity as it is massively dependent on the background, it is not invariant to translation or rotation, and perturbations of the same Euclidean distance have very different perceived distortion as demonstrated, for example, in Wang (2004).

One way to overcome these shortcomings of the euclidean distance is kernels, which allow us to translate an arbitrary notion of similarity into a euclidean distance. One interpretation of deep neural networks used, for example, by Simon et al. (2017) and Simon et al. (2018), is kernelized linear discriminant analysis. To this end, the whole neural network is the feature function of the kernel, and the last linear layer performs a logistic regression in the feature space.

Therefore, to explain a deep neural network, we can search for the closest training examples in this feature space. These examples can be presented to the user to explain of the learned invariances or to identify important features that all presented examples share. In particular, it is useful when understanding why the deep neural network failed to classify some inputs correctly.

Caruana et al. (1999) recommended this method for neural networks in a medical example. Since then, multiple improvements where recommended. For instance, Tschandl et al. (2020) present similar images from each class and Simon et al. (2017) and Simon et al. (2018) use not the whole representation but localized representations to find not entire images but regions of images that are similar according to the deep neural network. This progession is especially significant since we expect a good classifier to be invariant with respect to translation and rotation of objects.

This invariance leads to semantic areas of an image being detected in various image locations.

However, this explanatory method has some weaknesses. For example, the method only presents similar images and does not show why the images are similar. This point is key when using this method to understand a classifier, but is difficult when the images have multiple similarities or no obvious similarities. Therefore, the method can lead to ambiguous results that require a lot of further interpretation.

2.5 Background of Adversarial Examples

Adversarial examples were introduced for neural networks by Szegedy et al. (2013). They observed what they called blind spots in neural networks. They state:

> "Our main result is that for deep neural networks, the smoothness assumption that underlies many kernel methods does not hold. Specifically, we show that by using a simple optimization procedure, we are able to find adversarial examples, which are obtained by imperceptibly small perturbations to a correctly classified input image, so that it is no longer classified correctly." Szegedy et al. (2013)

This observation is the foundation of the definition of adversarial examples. For a given input image I, a corresponding adversarial example A_I with respect to a classifier F is an image that meets two conditions:

1. The difference τ between the example I and the adversarial example A_I is imperceptibly small,

$$A_I = I + \tau \quad \textbf{s.t.} \quad \tau \text{ imperceptibly small.} \tag{2.74}$$

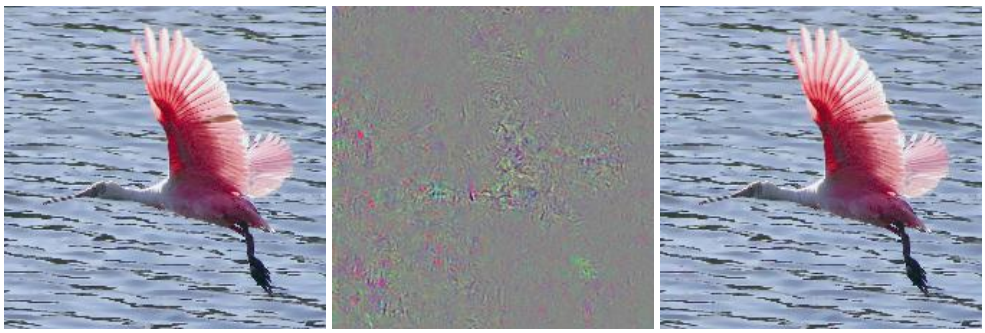
   We call τ the adversarial perturbation.

2. The example I is classified correctly by the classifier F as c^*, but the corresponding adversarial example A_I is classified as a different class c. Meaning

$$F(I) = c^*, \quad F(A_I) = c \quad \textbf{s.t.} \quad c^* \neq c. \tag{2.75}$$

A real-world example for such a combination of an image I, a corresponding adversarial image A_I and the adversarial perturbation τ can be found in Figure 2.2.

This observation is intriguing because it challenges three fundamental beliefs about deep neural networks. First, it seems to contradict to the impressive generalization performance of deep neural networks against random noise. The imperceptibly small perturbations seem not connected to any class and look like random noise. Second, it contradicts the common belief that deep neural networks are hierarchical feature extractors. This belief is a claim made by multiple authors Erhan et al. (2009); Olah et al. (2017). They state that the first layer of neural networks extracts basic features such as edges or colors, subsequent layers extract features



Class	Prob.
Spoonbill	0.9999
Flamingo	0.0001
Am. Egret	0.0000
...	...
Library	0.0000

Class	Prob.
Wall Clock	0.3068
...	...
Spoonbill	0.0003
...	...
Library	0.0000

Class	Prob.
Library	0.6289
...	...
Spoonbill	0.0152
...	...
Wall Clock	0.0000

Figure 2.2: *An example for an original example on the left, an adversarial perturbation in the middle (magnified) and the combined adversarial example on the right. The output of the classifier is listed below the images. The original image is correctly classified as a spoonbill, and the adversarial image is classified as a library. For this example, we used a ResNet50 classifier (He et al. (2016)) trained on ImageNet (Russakovsky et al. (2015)). We used the Projected Gradient Descent attack presented in Madry et al. (2017) to calculate the adversarial perturbation. Photo by luis rock from FreeImages*

comprised of these basic features, such as corners or basic shapes. Every higher layer then extracts features built of the features in the layer bevor until we end up with high-level features that detect object parts, for example eye shapes. This idea of a hierarchical classifier was used as motivation for other research like Simon and Rodner (2015) and discussed critically, for example, by Dong et al. (2017). The notion of adversarial examples is not compatible with this idea of a hierarchical feature extractor. The small perturbation used to change the prediction of the deep neural network is imperceptible. In particular, it does not change basic, easily perceptible features of an image, such as edges. Since these basic features are still intact, features that are simply combinations of these features should not change either. The observations, however, speak a different language. In Dong et al. (2017), the authors investigate the difference of neurons to images and corresponding adversarial images. They found that neurons in the first layers respond similarly to the original image and the image containing the adversarial perturbation. In contrast, the neurons in later layers respond vastly differently to the original and the adversarial image. The authors conclude that neurons from later layers do not react to high-level semantic features but just to more complex pixel patterns. Third, this seems to be a contradiction to the postulates of Niemann (1990) that are fundamental to many theoretical arguments in pattern recognition. A selection of these postulates states:

> **Postulate 2:** "A (simple) pattern has features that characterize its membership in a certain class." (Niemann, 1990)
>
> **Postulate 3:** "Features of patterns of one class occupy a somewhat compact domain of feature space. The domains occupied by features of different classes are separated." (Niemann, 1990)
>
> **Postulate 6:** "Two representations are similar if a suitably defined distance measure is small." (Niemann, 1990)

An imperceptibly small perturbation only changes the features that characterizes its membership in a certain class very little. Especially in tasks that are easy for humans such as the classification of photographs of everyday objects, the features that are important for the inference of the correct class are very perceptible. The influence an imperceptibly small perturbation can have on them is very limited. Consequently, according to the second postulate, the original and corresponding adversarial image contain the same characteristic features. According to the sixth postulate, the original image and the corresponding adversarial attack are close concerning a suitably defined distance measure. Further, since the features of patterns of one class occupy a somewhat compact domain that is separated from the domains occupied by features of different classes, such a small perturbation should not be able to change the classification of a pattern.

Since the existence of these adversarial examples challenge fundamental believes a lot of research has been conducted to shed light on how adversarial examples can be created, on why adversarial examples exist, how the above-mentioned contradictions can be resolved and how neural networks can be defended against such attacks.

2.5.1 Creation of Adversarial Examples

The main idea for creating adversarial examples is to calculate the gradient of the difference between the score of the target class and the original class depending on the input image. Then we look for the smallest perturbation in this direction that moves the image across the decision boundary. The classifier is a non-linear function, and the derivative of the original score-difference is only a linear approximation of these functions. Hence, most methods use an iterative approach where they make an initial guess and update it using the gradients at this initial guess. Multiple methods for the details on how to make initial guesses and how to update them are proposed in the literature, each of which has advantages and drawbacks. In this work, we introduce only some representing of the main ideas. These ideas include the original algorithm introduced in Szegedy et al. (2013) and its improvements created in Goodfellow et al. (2014b) and Madry et al. (2017). Further, we present the DeepFool algorithm presented in Moosavi-Dezfooli et al. (2016), the algorithm presented in Carlini and Wagner (2017) and the one-pixel attack described by Su et al. (2019).

The first algorithm to create adversarial examples is proposed by Szegedy et al. (2013), with the observation that adversarial examples exist. They propose to use the

L-BGFS algorithm, a memory-efficient variant of the Broyden-Fletcher-Goldfarb-Shanno algorithm (Fletcher (2013)). This algorithm uses the direction indicated by the initial derivative and then performs a line-search in this direction to find the perturbation of minimum size. Since the initial direction might not be optimal, this can lead to larger than necessary perturbations.

Goodfellow et al. (2014b) introduced the Fast Gradient Sign Method (FGS) to speed up the calculations. The idea of that method is to use the sign of the gradient as a direction and make a fixed size step in that direction. As such, the method is very fast but the resulting perturbations might be larger than necessary or might not lead to misclassification.

Another optimization technique to generate adversarial examples proposed by Madry et al. (2017) is the projected gradient descent. The main idea is to use the FGS algorithm iteratively. This change increases the quality of the results while also increasing the time spent for each example.

The DeepFool algorithm introduced in Moosavi-Dezfooli et al. (2016) approximates the decision boundary around the data point as a polyeder. It uses the derivative at the original image to find the direction of shortest distance to the edge of this polyeder. They find the smallest perturbation that should lead to a different classification in that direction. Afterward, if the classification has not changed, they repeat the process with the perturbed instead of the original image. While this leads to very small perturbations that consistently change the classification of the image, the algorithm is quite slow.

The idea presented in Carlini and Wagner (2017) is to replace relevant parts of the network that hinder backpropargation. The final decision function of a deep neural network is trained to look like a step function. Since step functions do not create gradients suitable for optimization, it gets replaced by a different function. The authors of Carlini and Wagner (2017) propose different replacements depending on the desired properties of the adversarial example.

A different approach to the ideas presented above is proposed in Su et al. (2019). The authors of that paper produce adversarial examples that differ only in one pixel from the original image. The authors do not use a version of gradient descent but differential evolution to find these adversarial perturbations.

Many more algorithms are proposed to create adversarial examples, for example, BIM (Kurakin et al., 2016), JSMA (Papernot et al., 2016b), Decision Tree Attacks (Papernot et al., 2016a), D-Patch (Liu et al., 2018b), Elastic-Net Attack (Chen et al., 2018), HCLU (Grosse et al., 2018), HopSkipJumpAttack (Chen et al., 2020), Newton-Fool (Jang et al., 2017), Shadow Attack (Ghiasi et al., 2020), SimBA (Guo et al., 2019), Square Attack (Andriushchenko et al., 2020), Wasserstein Adversarials (Wong et al., 2019), ZOO (Chen et al., 2017) and the decision-based attack (Brendel et al., 2017). These algorithms share the ideas mentioned above and present improvements or different compromises between runtime and quality of the resulting adversarial examples.

2.5.2 Intriguing Properties of Adversarial Examples

Another large field of research focuses on a better understanding of adversarial examples. Researchers have found several intriguing properties of adversarial examples. These properties make the problem of adversarial examples more interesting and allow us to reduce the number of possible reasons for adversarial examples. In the following, we briefly introduce the properties that, in our opinion, are most relevant to understanding the reason for adversarial examples.

2.5.2.1 Adversarial Examples Exist for Almost Any Kind of Data, any Task and Any Network Architecture

The first observation of adversarial examples is made by Szegedy et al. (2013) for a classification task on ImageNet (Russakovsky et al., 2015) data. In that dataset, real-world images have to be classified as one of 1000 classes, some of which are fine-grained, for example, different breeds of dogs, while others are more coarse, for example, airplanes and libraries. However, later researchers showed that the problem of adversarial examples is not limited to this kind of task. Goodfellow et al. (2014b) reveal that almost every task and example within a dataset is vulnerable to adversarial examples. The problem is further in no way bound to computer vision. Adversarial examples are also observed in other tasks, for example, natural language processing (Carlini and Wagner, 2018) or text (Ebrahimi et al., 2017). Further, the problem of adversarial examples is not exclusive to neural networks. For example, Dalvi et al. (2004); Lowd and Meek (2005) present ideas on robustness against an adversary for classic machine learning methods and Tatu et al. (2011) present an algorithm to derive adversarial examples for a classifier based on histogram of gradient features.

2.5.2.2 Adversarial Examples do not Resemble the Target Class

Another observation made, for example, by Goodfellow et al. (2014b) about adversarial examples is that neither the adversarial perturbation nor the resulting adversarial example visually resembles the target class. This observation is important. As demonstrated, for example, in Figure 3.2, we expect the classification of a neural network to change if we mix important features of a different class into the image. Especially if the target class has simple, decisive features, even a small perturbation adding this feature can be enough to change the classification decision of a neural network. However, the same happening without adding a clear feature of the target class is intriguing. We further visualized this effect in Figure 2.2.

Using the fact that adversarial perturbations do not change the correct label of the example, Goodfellow et al. (2014b) suggest using it to generate adversarial examples data augmentation. They claim that this data augmentation aids generalization.

One reason for this observation might be that deep neural networks have found clear, decisive features that are not obvious to humans. Since deep neural networks

have reached superhuman performance in many vision tasks, for example, object classification (Russakovsky et al., 2015) and in automatic skin lesion classification (Tschandl et al., 2019), it is fair to assume that deep neural networks have found meaningful features that are not obvious to humans.

2.5.2.3 Adversarial Examples are Robust to Random Noise

Even though adversarial examples result from an exact calculation process, the adversarial perturbations look like random noise and do not resemble the target class (comp. Figure 2.2). This property is intriguing, as neural networks have proven to generalize well against noise. A natural attempt to mitigate the problem is, hence, to either add gaussian noise to the adversarial example to obfuscate the adversarial perturbation (Tabacof and Valle, 2016) or to use methods that have proven to be effective against other kinds of noise, for example, denoising autoencoders (Gu and Rigazio, 2014). Tabacof and Valle (2016) compare the robustness of adversarial and original images to noise. They generate pairs of clean and adversarial examples. Then they add random noise of increasing norm to both the adversarial and original image. During this process, they measure whether the classifier's prediction changes due to the noise. They find that the noise needed to change the classification of an original example is of a similar magnitude as the noise needed to change the classification of an adversarial example. Even though the original examples are more robust, they conclude that the adversarial examples are by no means isolated points but populate dense areas. Bai et al. (2017); Gu and Rigazio (2014) use an autoencoder to denoise the adversarial examples. While they found reasonable success against an adversary not aware of this defense, they found it easy for an adversary to create adversarial examples for this combination of two deep neural networks. Similarly, Li and Li (2017) and Lu et al. (2017) try to evaluate the statistics and distributions in later layers of the deep neural network to identify adversarial examples. However, while they show significant differences in the distribution of adversarial and original examples, these differences are not enough to decide whether a single example is adversarial or themselves vulnerable to adversarial attacks. Further analysis about the connection between the robustness against adversarial examples and random noise can be found in Fawzi et al. (2015, 2018, 2016); Gilmer et al. (2019); Rozsa et al. (2016a); Stutz et al. (2019); Su et al. (2018); Tsipras et al. (2018). In summary, adversarial examples are not isolated and, therefore, simply adding noise can not reliably recover the correct classification.

2.5.2.4 Adversarial Examples are Transferable Between Different Networks

As described in the previous paragraph, the threat of adversarial examples can not be reliably mediated by adding random noise or using denoising autoencoders. However, most attack algorithms rely on specific model parameters to calculate a gradient. To this end, a natural strategy to defend a neural network is to hide these parameters and not do allow an attacker to calculate a gradient. Unfortunately, this

defense did not prove to be effective. Different researchers found that adversarial examples derived for one classifier are likely to fool other classifiers. For example, Szegedy et al. (2013) find that an adversarial example created for a handwritten digit classifier trained on one half of the MNIST dataset (LeCun, 1998) can be transferred to the same model trained on the other half. Further, researchers have demonstrated that adversarial perturbations can transfer between different examples, forming universal adversarial perturbations (Metzen et al., 2017; Moosavi-Dezfooli et al., 2017). Adversarial examples transfer between deep neural networks with different architectures (Li et al., 2019a; Tramèr et al., 2017b) or even from more traditional classifiers to deep neural network classifiers (Papernot et al., 2016a). In conclusion, adversarial attacks can be transferred between classifiers. Hence, hiding the parameters, the architecture and the data on which the classifier is trained does not pose a valid defense. The transferability of adversarial examples further demonstrates that adversarial examples are not due to overfitting effects that over-emphasize single examples, as such deficits would not be transferable to classifiers trained on other datasets.

2.5.2.5 The Curvature of the Decision Boundary Near Adversarial Examples is Positive and the Direction of Adversarial Perturbations is Similar Across Multiple Examples

If a perturbation changes the classification of an image, it has to "push" the example over the decision boundary. Multiple researchers observe that the decision boundary is negatively curved (Moosavi-Dezfooli et al., 2019; Tramèr et al., 2017b) at the adversarial examples. This observation leads to investigations between the geometry in the feature space and adversarial examples (see, for example, Gilmer et al. (2018b); Stutz et al. (2019)). The fact that this curvature is negative is especially surprising in high dimensions, as the area close to the decision boundary is much larger on the side of positive curvature than negative curvature.

2.5.2.6 The Correlation Between Robustness and Accuracy

Another property of adversarial examples that has been investigated is the connection between accuracy and adversarial robustness. While early research like Rozsa et al. (2016a) concluded that accuracy and robustness are positively correlated, newer research, for example, Su et al. (2018); Tsipras et al. (2018), find that robustness and accuracy might conflict.

2.5.3 The Threat Level due to Adversarial Examples

Adversarial examples seem like a serious real-world threat, given the difficulty of defending deep neural networks against them. This fact is further corroborated by the research of, for example, Eykholt et al. (2017), Metzen et al. (2017) and Sharif et al. (2019). Their research suggests that adversarial attacks are robust enough to be carried out in the wild. Eykholt et al. (2017) create adversarial stickers

that change the classification of a street sign even if photographed from multiple angles and distances. Metzen et al. (2017) suggest a method to create adversarial patterns that can be added to any image of an automobile camera and removes the pedestrians from the segmentation map of the image. Sharif et al. (2019) present patterns that can be printed on cloth or accessories like glasses that can doge face recognition or detection by neural networks. However, Gilmer et al. (2018a) describe, it is difficult to define a concrete thread scenario, where an adversarial attack against an automatic classifier would be the easiest option for an adversary. If we want to hinder the recognition of a street sign, it might be easier to either remove or exchange it. If we can add noise to the image captured by a camera in an autonomous car, we can simply replace the entire image instead of adding imperceptible noise and, finally, an adversary that wants to doge face recognition might simply wear a mask instead of glasses with an adversarial pattern.

Hence, our main interest in adversarial examples is not driven by security but by curiosity. The existence of adversarial examples does not agree with some common assumptions made when working with deep neural networks and understanding why they exist. It will help us develop better neural network classifiers.

2.5.4 Theories on Why Adversarial Examples Exist

Given the intriguing properties of the adversarial examples, they reveal that we have to reevaluate our knowledge of deep neural networks. To this end, multiple researchers have proposed theories on why adversarial examples exist. As they proposed theories of how deep neural networks learn from data that include the existence of adversarial examples. These theories help us to better understand deep learning as a whole better. In the following, we introduce some of the most important theories on why adversarial examples exist. Note that, just because a theory can not explain all properties of adversarial examples, it does not have to be wrong. It is, instead, very likely, that multiple phenomena cause the effect of adversarial examples, and adversarial examples caused by different phenomena will have different properties.

2.5.4.1 Pockets of Low Probability

The original publication, Szegedy et al. (2013), that introduced adversarial examples, also proposed the first explanation on why they might exist. They propose that adversarial examples form pockets of low probability that are dense in the feature space, similar to how the rational numbers are dense in the real numbers form a dens but zero-probability set. In other words, adversarial examples are an overfitting effect, where a classifier approximates the true decision function using an approximation with higher complexity than the true decision function. This explanation is capable of explaining why adversarial examples exist close to every example in the data set but do not occur naturally in the training or test set.

It is only half of an explanation, as it is unclear why deep neural networks would create such pockets in the first place. However, this question is central to

understanding the properties of adversarial examples. For example, the fact that adversarial perturbations transfer between different neural networks trained on different datasets and between different examples in the same dataset indicates that the position of these pockets is far from random.

Further, small isolated pockets would not be robust to random noise. This theory can explain the observation by Tabacof and Valle (2016). The same holds for the observation that the curvature is negative. If adversarial examples formed small pockets, we would expect the curvature of the decision boundary that we cross from outside to inside to be positive. But it was consistently found to be negative by, for example, (Moosavi-Dezfooli et al., 2019; Tramèr et al., 2017b).

2.5.4.2 Networks Only Learn a Low Dimensional Manifold and are Random Outside of It

It is known that real-world images only form a low dimensional manifold inside the pixel space $[0,1]^m$. One proposed theory is that deep neural networks only make viable predictions for the data on this manifold. In contrast, the predictions outside this manifold mainly rely on the random initialization.

While a lot of research indicates that the neural network behaves differently inside and outside the data manifold (Stutz et al., 2019), the random initialization can not explain the properties of adversarial examples. Even though this theory explains why the curvature is negative, it does not explain why adversarial examples are transferable between different neural networks or between different examples.

2.5.4.3 Diminishing Learning Effect of Positive Examples

This theory was proposed by Rozsa et al. (2016b). They suggest that the deep neural network mainly focuses on wrong classifications during training. To this end, the authors claim that neural networks build homogenous regions around falsely classified examples. In contrast, correctly classified examples do not contribute as much to the loss, and, hence, no homogenous regions are built around them.

Even though this theory does not attribute the existence of adversarial examples to overfitting, the resulting classifier is similar to the classifiers proposed by the two previous theories. The classifier will work well in regions where many datapoints are present and will be basically random in other areas. Therefore, this theory can explain the same properties of adversarial examples like the ones above.

2.5.4.4 Networks are Too Linear

An alternative to the abovementioned theory of small pockets that attributes the existence of adversarial examples to overfitting of a classifier due to high complexity is presented by Goodfellow et al. (2014b). The authors propose that the reason for adversarial examples is that the non-linear true decision function can not be approximated well enough by the piece-wise linear function represented by the

deep neural network. Hence, they attribute the existence of adversarial examples to an underfitting effect.

Many of the observations corroborate this claim. First, multiple algorithms that have proven effective to generate adversarial examples, for example, the L-BFGS algorithm (Szegedy et al., 2013), are linear. Second, since different networks approximate the decision boundary similarly, this theory can explain why adversarial examples are transferable between neural network classifiers. Further, since the dimensionality of the feature space is high compared to the number of examples, we expect every example to lie close to the decision boundary.

However this theory can not explain why adversarial perturbations transfer between different examples within a dataset, as observed, for example, by Metzen et al. (2017) and Moosavi-Dezfooli et al. (2017). Underfitting the decision boundary would lead to pockets at both sides of the decision boundary. Hence, it is also difficult to explain why we find the curvature negative.

2.5.4.5 Adversarial Examples are a Natural Consequence of Imperfect Generalization

Similar to the previous theory of the neural network being too linear, this theory also attributes the existence of adversarial examples to an underfitting problem of the classifier. Similar to the abovementioned theory, the idea is that a neural network only approximates the true decision boundary and, hence, will create pockets next to the decision boundary. The idea of how this creates adversarial examples is the same as in the previous theory, and, hence, it explains the same observations. Additionally, however, Gilmer et al. (2018b) and Gilmer et al. (2019) present more evidence for this theory. They present empirical evaluations and calculations in toy examples to corroborate this theory.

A significant consequence of this theory would be that the accuracy and the adversarial robustness are correlated. Multiple researchers investigated the connection between accuracy and adversarial robustness. While early research like Rozsa et al. (2016a) concluded that accuracy and robustness are positively correlated, newer research, for example, Su et al. (2018); Tsipras et al. (2018), find that robustness and accuracy might conflict.

2.5.4.6 Boundary Tilting Perspective

Another theory on why adversarial examples exist is the boundary tilting perspective introduced by Tanay and Griffin (2016). The main idea behind this theory is that the decision boundary is underdefined due to the sparsity of the data. The locally linear decision boundary is tilted randomly along the axis where the data has little variation. To corroborate the claim that random tilting takes place and can lead to adversarial examples, the authors present two simple examples using linear support vector machines. The tilting of their decision boundary can be controlled with the regularization parameter of the support vector machine. In these experiments,

the misclassifications that result from a tilted decision boundary visually resemble adversarial examples.

The high dimensionality of the data should ensure that there is a decision boundary close to every example in most tasks. This fact explains why there is an adversarial example close to every example. Since the decision boundary is in only one direction from the example, random noise that expands in all directions will rarely cross the decision boundary. Hence, adversarial examples will be robust to random noise. This theory can also explain why adversarial examples transfer between examples. Since the adversarial perturbations are along the axis of low variance in the data, the direction will be the same for different examples and classifiers. However, this theory can not explain the observations on the curvature of the decision boundary.

3 | Adversarial Examples

3.1 A Definition for Adversarial Examples

Despite the impressive amount of research conducted on adversarial examples (see Section 2.5), there is still disagreement on some fundamental aspects. One of these aspects is the definition of adversarial examples. In Szegedy et al. (2013), the authors state that adversarial examples are "imperceptibly small perturbations to a correctly classified input image, such that it is no longer classified correctly." To turn this description of adversarial examples into a mathematical definition, one has to give a formal definition for its important parts. Authors use a variety of definitions that focus on different aspects of this initial description. In this section, we, first, introduce a selection of these definitions and discuss the selected focuses. Second, we propose our definition of adversarial examples and compare it to the definitions from the literature.

3.1.1 First Definition

The first definition is proposed by Szegedy et al. (2013). In the notation from Section 2.5, they define adversarial examples as any feasible solution to the optimization problem

$$\min ||\tau||_2 \quad \text{s.t.} \tag{3.1}$$
$$F(I+\tau) = c \neq c^* \tag{3.2}$$
$$I + \tau \in [0,1]^m. \tag{3.3}$$

This definition aims to find a perturbation that changes the output of the classifier F (3.2) under the restriction that the resulting input should still be an image (3.3). This definition, however, is far from the original description of adversarial examples. It does not capture the idea of imperceptibly small perturbations, as a feasible solution to this problem could be arbitrarily large. Therefore, it does not capture the intriguing part of adversarial examples. In particular, every correctly classified example I_c of class c would be an adversarial example to every correctly classified example I_{c^*} of class c^*. Therefore, $\tau = I_c - I_{c^*}$ is a feasible solution since

$$F(I_{c^*} + \tau) = F(I_{c^*} + I_c - I_{c^*}) = F(I_c) = c \tag{3.4}$$

and

$$I_{c^*} + \tau = I_{c^*} + I_c - I_{c^*} = I_c \in [0,1]^m \tag{3.5}$$

hold. Hence, we follow Gu and Rigazio (2014), which states that:

> "One could always engineer an additive noise at input to make the model misclassify an example, and it is also a problem in shallow models such as logistic regression Szegedy et al. (2013). The question is how much noise is needed to make the model misclassify an otherwise correct example. Thus, solving the adversarial examples problem is equivalent to increasing the noticeability of the smallest adversarial noise for each example." (Gu and Rigazio, 2014)

3.1.2 Restricting the Norm of Tau

Various researchers, for example, Kurakin et al. (2016) and Tramèr et al. (2017a), have used stricter definitions for adversarial examples since the previous definition is too loose and classifies too many input images as adversarial examples. The main approach in this direction is to limit the norm of the perturbation τ leading to the optimization program

$$\min ||\tau||_2 \quad \text{s.t.} \tag{3.6}$$

$$F(I+\tau) = c \neq c^* \tag{3.7}$$

$$I + \tau \in [0,1]^m \tag{3.8}$$

$$||\tau|| < \epsilon. \tag{3.9}$$

However, neither ℓ_p-norm is considered a good measure for imperceptibility, as discussed in Wang (2004) and Zhang et al. (2018b). While most researchers agree that a very small norm of a perturbation leads to imperceptibility, the threshold of which perturbation is still imperceptibly small is highly subjective. Different authors have proposed different values. For example, Madry et al. (2017) limit the infinity norm $||\tau||_\infty$ to 0.3 for an MNIST (LeCun, 1998) example and to $||\tau||_\infty < 8$ for an experiment on CIFAR (Krizhevsky et al., 2009). But authors not only disagree on the threshold for different applications but also report different values for ϵ on the same task, and some authors, for example, Kurakin et al. (2016) and Tramèr et al. (2017a), report multiple thresholds for one task. These discrepancies demonstrate the difficulty in measuring imperceptibility.

Further, perceptibility also depends on the user. Outside of computer-vision applications, any perturbation might be imperceptible for non-experts. Especially in unusual data that requires expert knowledge, such as medical data or climate data, it will depend on the user's expertise whether a perturbation is perceptible.

Therefore, most authors disregard the perceptibility and focus on the notation of *small*. However, while the notation of *small* is more objective than the notation of perceptibility, defining what constitutes a *small* perturbation is still not straightforward. To this end, we present an example to illustrate that the size of a *small*

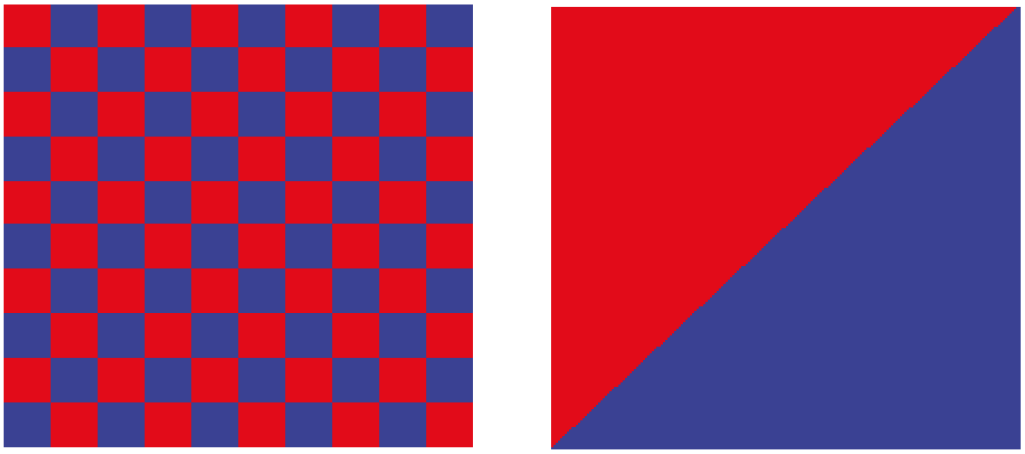


Figure 3.1: *The ground truth for the class of an input uniformly distributed on* $[0,1]^2$. *The first class is marked red and the second blue. On the left, the average norm of the smallest perturbation needed to change the label of an input is* 0.0167, *while on the right, it is* 0.25.

Figure 3.2: *Linear interpolation between an image of a car and an image of a horse from the CIFAR-10 data set (Krizhevsky et al., 2009). The parameter* λ, *defined in* (3.10), *increases by 0.1 in every image, starting at zero on the left and ending at one on the right.*

perturbation depends on the data set and can be different even if the dimension, the scaling, the number of classes, and the area per class are the same in two data sets.

For an adversarial example $I+\tau$ to be close to the corresponding input example I, we need the adversarial perturbation τ to be of small norm $||\tau|| < T$. The threshold T is not only subjective but also depends on the data set. Obviously, T will be bigger if the input data is in the range $[0,255]$ than if the input data is in the range $[0,1]$. Further, the data manifold of real images is often much smaller than the input space. Small perturbations relative to the data manifold will be smaller than small perturbations relative to the input space. Additionally, the relative position of examples in the feature space is important to determine what perturbation can still be considered small. As described above, a large enough perturbation will always be able to change the prediction of a classifier. Therefore, to count as small, we want a perturbation to be small compared to a perturbation that changes the classifier's prediction because the content of the image has changed so much, that a human expert would change their prediction, too. However, the size of a perturbation that is supposed to change the label depends not only on the number of classes and the dimensionality of the data but also on the concrete distribution of the data in the feature space. For example, consider the two binary classification tasks displayed in

Figure 3.1. The inputs for both tasks are uniformly distributed in two dimensions on the interval $[0, 1]$. The ground truth for the classification label is indicated by color. The blue areas contain examples of class one, and the red areas contain examples of class two. Both these classes occupy the same area in the input space in both tasks. While in the first case, in the left of Figure 3.1, a perturbation of norm 0.05 is always enough to change the class label in the second case, on the right, it takes a perturbation of norm up to 0.707. This example demonstrates that T should be chosen differently, even though the dimensions, the scaling, the number of classes, and the area per class are the same in both tasks. Furthermore, it demonstrates that it is impossible to find a data set-independent measure for small perturbations.

In this work, we make the notation of small perturbations more objective. We compare the adversarial perturbations to the following non-adversarial perturbations to determine what should constitute a small perturbation. We start by introducing a method that produces a perturbation suited to change the decision of a classifier in a non-intriguing way. We use the linear interpolation

$$I_{c^*} + \tau_\lambda = \lambda I_c + (1 - \lambda) I_{c^*} \tag{3.10}$$

between an example I_c from the target class and the original example I_{c^*}. The perturbation τ_λ is, hence, given by

$$\tau_\lambda = \lambda I_c - \lambda I_{c^*} \tag{3.11}$$

for $\lambda \in [0, 1]$.

Note that for $\lambda = 0$, the perturbed image $I_{c^*} + \tau_\lambda$ equals I_{c^*}, and for $\lambda = 1$, the perturbed image $I_{c^*} + \tau_\lambda$ equals I_c. Hence, for large enough λ, the perturbed image should be labeled as class c, as it resembles the target image more than the original image. For $\lambda \in (0, 1)$, the perturbed image is a linear interpolation between I_{c^*} and I_c, as displayed in (3.10). An example of such interpolation can be viewed in Figure 3.2.

While this looks very similar to the definition of adversarial examples, it misses one of the main properties that make adversarial examples intriguing. The adversarial perturbation and the resulting adversarial example do not resemble the target class. Since the perturbed image following this process is a linear interpolation between the input and an image of the target class, it will resemble the target class.

We define a perturbation as *small* if it is small compared to the smallest τ_λ that can fulfill

$$F(I_{c^*} + \tau_\lambda) \neq c^*, \tag{3.12}$$

which we calculate through solving the optimization problem

$$\tau^0 := \mathbb{E}_{I_{c^*}, I_c}\left[\min_{\lambda \in [0,1]} ||\tau_\lambda|| \quad \textbf{s.t.} \quad F(I_{c^*} + \tau_\lambda) \neq c^*) \right]. \tag{3.13}$$

We incorporate this idea of *small* into the definition of adversarial examples to get Definition 1.

Definition 1. *For a classifier F and an input example I_{c^*} that is classified as $F(I_{c^*}) = c^*$, an adversarial perturbation τ is given by every feasible solution to*

$$\min_{\tau:||\tau||<\eta||\tau^0||} ||\tau|| \quad \textbf{\textit{s.t.}} \quad F(I_{c^*} + \tau) = c \neq c^*.$$

The data point $I_{c^*} + \tau$ is called an adversarial example. *If we construct the adversarial example for a fixed class c, we call it a* targeted adversarial example *and c the* target class.

This definition makes the notation of *small* independent of the data set. It reduces the task of selecting an individual threshold for each task to selecting one parameter η through the use of τ^0. Of course, calculating the expectation in (3.13) is hard to calculate over large datasets. However, this problem can easily be met by calculating it only on a representative subset of the dataset. To show that τ^0 is useful to make the notation of a small perturbation objective and independent from the dataset, we conduct experiments in Section 3.3.

3.1.3 Restricting the Perturbation to be Imperceptible

Some papers try to make the idea of imperceptibly small perturbations objective by focusing on imperceptibility. However, the perceptual similarity is in the eye of the beholder, and the concept of imperceptibility is challenging to extend to non-image signals like time series, which only experts might be able to tell apart in the first place.

We will first elaborate on why it is difficult to measure imperceptibility. A perturbation is imperceptible if the perturbed image perfectly resembles the original image. We argue that no simple function can measure resemblance objectively. We show that subjective measures by humans depend on the data set, and we give examples for situations where humans might miss resemblance. Afterward, we propose a method to objectively measure resemblance on any data set. We propose to train a second neural network on the adversarial examples and check whether the patterns it finds in adversarial examples with the same target are useful to identify the class in clean images.

We start with three reasons why the task of identifying resemblance is challenging:

First, it is impossible to solve this problem with a simple metric. Measuring visual resemblance is an open problem, and as described by Wang (2004), there are multiple arguments not to assume that a simple metric can solve it. The most successful way to measure visual resemblance is to use features extracted from neural networks, as described by Zhang et al. (2018b). This way is, however, problematic in the setting of adversarial examples.

Second, the visual resemblance between different images depends on the data set. Compare a data set of real-life photographs and a data set of medical scan images. The first kind of data is familiar to most observers, and they will easily detect perturbations and might be able to identify a resemblance. Contrarily, the second

kind of data is unfamiliar to most observers, and they might miss even significant perturbations and judge visual resemblance differently than domain experts. The same is true for data sets of different granularity. Two species of birds might look similar to many people, even though the difference is distinct to a domain expert. Therefore, we cannot rely on humans if we seek a data set-independent measure for visual resemblance.

Third, one strength of deep neural networks is the automatic feature selection. These features might not be interpretable or apparent for humans. To illustrate, we describe the following example inspired by Tsipras et al. (2018) of a binary classification task in which humans and automatic classifiers might prefer different features. The two classes of the classification task are denoted by $y = 1$ and $y = -1$. For every instance $\mathbf{x}$ of these classes, the features are distributed as follows

$$\begin{aligned} x_1 &= \begin{cases} y & \text{w.p. } 0.9 \\ -y & \text{w.p. } 0.1 \end{cases}, \\ x_2, \ldots, x_{31} &\sim \mathcal{N}(0.1y, 1). \end{aligned} \tag{3.14}$$

In this setting, a human might consider the feature x_1 to be most relevant and especially to be more relevant than the other thirty features since the sign of feature x_1 agrees with the sign of y in 90% of cases and every one of the other thirty features has only a probability of 54% to share a sign with y. However, the feature that is the sum over all thirty of these features is of the same sign as y in more than 99.8% of the cases

$$\mathbb{P}\left(\text{sign}\left(\sum_{i=2}^{31} x_i\right) = y\right) > 0.998 \tag{3.15}$$

and might, hence, be preferred by an automatic system.

A more applied example was presented by Lapuschkin et al. (2019). They used spectral relevance analysis to understand the behavior of a classifier on horse images from the PASCAL VOC data set (Everingham et al., 2010). They found many of these images have a source tag in the bottom left corner. Suppose an adversary aims to perturb an image to be classified as a horse by adding a light spot to the bottom left corner. At first, it might seem that this has no visual resemblance to a horse. However, it is not a random perturbation but resembles many horse images in the dataset and is, hence, a feature that will generalize well across large parts of this specific data set.

These examples demonstrate that neural networks might use features that humans do not consider and that we have to be careful not to disregard features presented by neural networks as meaningless just because the resemblance to the target class is unobtrusive at first.

It is difficult to ensure that a perturbation is imperceptible. Researchers claim that perceptibility is linked to the semantic concept of the object that should be classified. For example, if we want to classify an object in front of a background, then a perturbation on the object is more perceptible than a perturbation on the

background. To this end, researchers have suggested different solutions.

One solution to create adversarial perturbations that are imperceptible is rotations. In classifying real-world images, a rotation of an image should not affect the correct label. Hence, this perturbation in the relevant feature space is imperceptibly small, even though it is large in the pixel space. This idea was first suggested by Goodfellow et al. (2014b) and further explored, for example, by Engstrom et al. (2019).

A different solution was proposed by Hosseini and Poovendran (2018). They convert images into the HSV color model and then randomly shift the hue and saturation components while keeping the value component fixed. They justify this with the "shape bias" in the human cognitive system.

Rozsa et al. (2016b) replace the ℓ_p-norm with the perceptual adversarial similarity score (PASS). This score is given by the structural similarity (Wang, 2004) between an image and its adversarial counterpart. Since this score better corresponds to the human perception, a small structural similarity will better correspond to imperceptibility than a small ℓ_p norm.

Other authors reach imperceptibility by adding the noise to areas that do not contain the relevant part of the image. Either by perturbing only parts of the image outside of the bounding box of the object that should be classified (Luo et al., 2015) or, for example, by perturbing only the lowest of the RGB values of each pixel (Carlini and Wagner, 2017).

Most of these advances follow a similar idea. They try to measure the distance between the clean and the adversarial example not in the pixel space. Instead, they define a suitable space where the distance corresponds to the perceptible distance. As discussed above, the ℓ_p-norm is not suitable to measure perceptible distance (Wang, 2004). However, Zhang et al. (2018b) state that the later layers of deep neural networks are unreasonably effective for capturing the perceptive distance. Hence, in contrast to the abovementioned works, we use deep neural networks to measure the perceptive distance.

3.2 A Quantitative Score for the Perceptibility of Adversarial Perturbations

In this section, we derive our score $\mathfrak{R}$ of adversarial robustness. We focus on whether a perturbation is imperceptible, which is an important property of adversarial examples. Following the research mentioned above, a perturbation is imperceptible if it does not change any suitable feature to distinguish the classes in the classification task. However, it is difficult to decide whether a feature is suitable to distinguish between classes. In particular, since deep neural networks have reached superhuman performance in some image classification tasks, see, for example, Russakovsky et al. (2015) and Tschandl et al. (2019), a perturbation that seems meaningless to a human might still be useful. Further, suppose we created adversarial examples for a deep neural network. We obviously cannot use the same neural network to determine

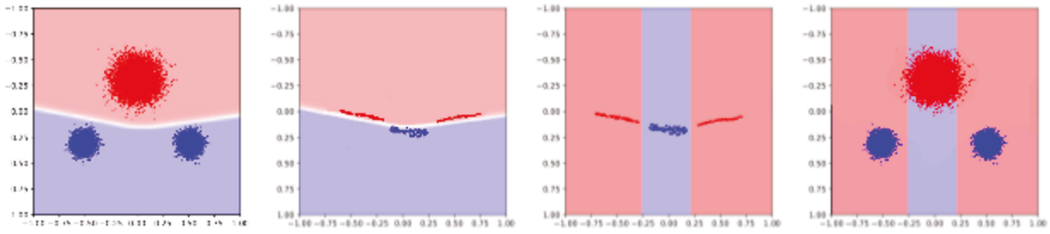


Figure 3.3: *We show the four steps of our approach. We start with a classifier. We calculate the closest input that is classified differently. We train a new classifier on this new set of inputs. We evaluate the new classifier on the original test set.*

whether the example has been changed in a meaningful way. The created example will fool the deep neural network whether it has been changed meaningful or adversarial. We, further, cannot use a different neural network because adversarial examples are transferable, as described in Section 2.5.

To solve this problem, we use learnability. Let I be an example of class c^* and $I + \tau$ a new example classified as class c different than c^*. If the perturbation τ is meaningless with regard to the classification, or if the new example $I + \tau$ is adversarial, it should not be possible to learn how examples of class c look from image $I + \tau$. On the other hand, if the perturbation is meaningful in the corresponding feature space, or in other words, if the perturbation τ is not adversarial, the new image should contain the relevant features, which the deep neural network uses to recognize images of class c and, therefore, these features can be learned from the perturbed image. As a consequence, we can determine whether a perturbation is adversarial by checking if we can learn the original task from the resulting examples.

Unfortunately, this idea does not allow us to determine which examples are adversarial. The reason is that we cannot train a classifier on a single example but a set of examples. Hence, we only determine whether a set of examples contains adversarial examples or examples that are perturbed in a way that changes the correct label of an image.

We propose the following algorithm to measure resemblance between adversarial examples and their target class: We generate one adversarial candidate for every instance in the training set with a random target label. We create the *adversarial set* from the original data set by replacing every image I with the adversarial image $I + \tau$ and every label to c other than the original class c^*. We display some examples from the adversarial set for the Fashion-MNIST data set (Xiao et al., 2017) in Figure 3.4.

If the images in the adversarial set have no resemblance to the target class, as which they are labeled in the adversarial set, then these images and their labels have nothing in common. The set should behave similarly to a randomly labeled set described, for example, by Zhang et al. (2016). The prediction performance of a neural network trained on a randomly labeled set does not generalize to a test set. If, contrarily, the perturbed images in the adversarial set resemble the target classes, as which they are labeled, a neural network can extract these meaningful features and generalize well on a test set, similar to a neural network trained on the original

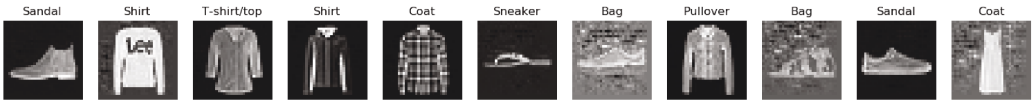


Figure 3.4: *Examples from the adversarial set created for the Fashion MNIST data set. The images are adversarial images for the Fashion MNIST data set to the target displayed over the images, which are the labels of these perturbed images in this adversarial set.*

training set.

The main steps of our method are presented in Figure 3.3. One of the central properties of adversarial examples is that there is no visual resemblance between the original image, the adversarial example and the adversarial perturbation. Hence, we need to ensure that the adversarial set behaves like a randomly labeled set. In practice, we will check that it behaves more similarly to a randomly labeled set than the original set. To this end, we train two identical neural networks. We train the first network F_O on the original training set and the second network F_A on the adversarial set. Afterward, we use the original test set to compare the generalization performance of the two networks. As a measure for generalization performance, we compare the confusion matrix of the neural network on the test set.

Formally, we compare the two hypotheses:

H_1 The confusion matrix of the second neural network F_A is identical to the confusion matrix expected from a neural network trained on randomly labeled data. This similarity indicates that the neural cannot learn the task from the adversarial set. Consequently, the labels of the adversarial set are not related to the perturbed images.

H_2 The confusion matrix of the second neural network F_A is identical to the confusion matrix of the first neural network F_O. The neural network trained on the adversarial set generalizes, and a neural network trained on the original data set. Consequently, the labels of the adversarial set are related to the perturbed images.

We can use the confusion matrix K of the neural network F_A on the test set as evidence to find the likelihood of these hypotheses using Bayes' theorem. To confirm that the perturbed images in the adversarial set are, indeed, adversarial examples, we want H_1 to be more likely than H_2. We assume that both hypotheses are equally likely a priori and calculate

$$\mathfrak{R} = \log \frac{p(\mathrm{H}_1 \mid K)}{p(\mathrm{H}_2 \mid K)} = \log \frac{p(K \mid \mathrm{H}_1)p(K)p(\mathrm{H}_2)}{p(K \mid \mathrm{H}_2)p(K)p(\mathrm{H}_1)} = \log \frac{p(K \mid \mathrm{H}_1)}{p(K \mid \mathrm{H}_2)}. \tag{3.16}$$

We accept hypothesis H_1, if $\mathfrak{R} > 0$.

3.3 Experiments

In this section, we present three experiments. First, in Section 3.3.1, we show empirically that the notation of "small" depending on τ^0 better allows us to compare the adversarial robustness on different datasets than ε in the original definition (3.9). Second, in Section 3.3.2, we show that our score $\mathfrak{R}$ agrees with the literature in differentiating between adversarial and non-adversarial examples. Third, we show that the adversarial robustness of a classifier is correlated to the modality of individual classes' distributions in the training set.

3.3.1 Demonstrating that τ^0 is Dataset Agnostic

We conduct experiments to demonstrate that our proposed method helps to make adversarial robustness more comparable across datasets. In the first experiment, we show that τ^0 can be calculated on a subset of the data. In the second experiment, we demonstrate that the same dataset under different transformations still has similar adversarial vulnerability if normalized by τ^0.

As a second experiment, we demonstrate that τ_0 makes the definition of adversarial examples robust against transformations of the dataset such as rescaling or a smaller data manifold embedded into a bigger space. We show that the parameter η in (3.13) captures the choice for *small* perturbations better than the original parameter $||\tau||_2$.

3.3.1.1 Efficient Computation of τ^0

For the first experiment, we test which fraction of the dataset is needed to determine τ^0 correctly. We test this for four datasets, namely MNIST (LeCun, 1998), Fashion-MNIST (Xiao et al., 2017), CIFAR10, and CIFAR100(Krizhevsky et al., 2009). The test sets to all these data sets contain 10^4 examples. The number of possible combinations of data points from different classes is, hence, $9 \cdot 10^7$ for MNIST, Fashion-MNIST and CIFAR10 and $9.9 \cdot 10^7$ for CIFAR100. We calculate τ^0 on different fractions of these combinations. The results for Fashion-MNIST can be seen in Figure 3.5. The results for the other datasets look similar.

As is expected because of the central limit theorem, the variance of the estimate for τ^0 gets smaller with the square root of the number of data points considered. We observe that the variance of the estimates decreases as proposed in the central limit theorem. However, for less than 10^{-4} of the examples, is the non-independence of the data problematic. Hence, after evaluating 10^{-4} of the data, one can use the variance on that subset to estimate how many samples are needed to reach a specific accuracy needed for a task. The values of τ^0 for the different datasets can be found in Table 3.1.

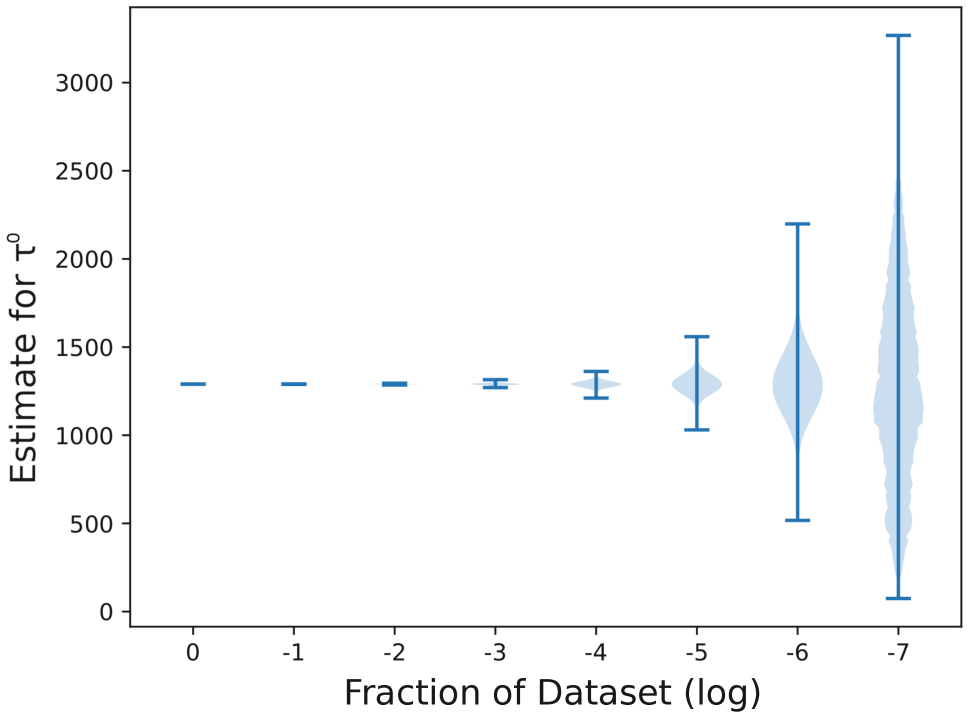


Figure 3.5: *The determination of τ^0 on the Fashion MNIST dataset from only a fraction of the data given on a log-scale to base ten. As expected, the variance is smaller when evaluating τ^0 on bigger fractions of the dataset. For fractions larger than 10^{-4}, the variance of the estimate follows the central limit theorem.*

Table 3.1: *The value of τ^0 for different datasets, as well as the standard deviation σ calculated from 10^{-4} of all examples*

Data Set	τ^0	σ at 10^{-4}
MNIST (LeCun, 1998)	1166.16	$\pm$32.32
Fashion-MNIST (Xiao et al., 2017)	1290.01	$\pm$55.33
CIFAR10 (Krizhevsky et al., 2009)	1705.81	$\pm$68.11
CIFAR100 (Krizhevsky et al., 2009)	1442.91	$\pm$75.811

3.3.1.2 Demonstrating that τ^0 is Independent of Transformations

Further, we conduct two experiments to demonstrate that this notation of small is robust against transformations of the dataset that do not affect the inner structure of the dataset and, hence, should not influence the adversarial robustness.

The first experiment we conduct for this purpose is a simple rescaling. We rescale the Fashion-MNIST datasets and find the adversarial robustness given by the success rate of the DeepFool algorithm developed by Moosavi-Dezfooli et al. (2016) for different values of η. We create the adversarial example with the smallest

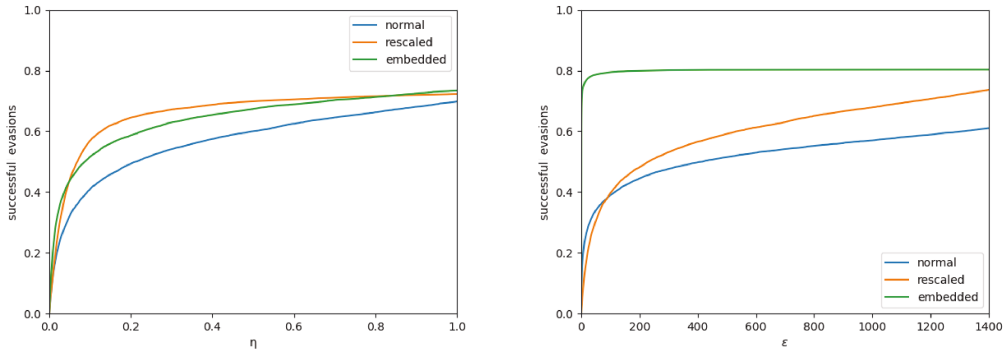


Figure 3.6: *The fraction of successful adversarial attacks with a perturbation of at most $\eta\tau^0$ in the Fashion-MNIST data set. We display, in blue, the values for the original dataset, in orange, the values for the dataset rescaled to the interval $[0,1]$, and, in green, the dataset linearly transformed by a random transformation and embedded into a bigger input space (32×32). On the left, we plot the fraction of sucessful attacks against η. On the right, we plot the fraction of sucessful attacks against ε, the maximal norm of the adversarial attacks.*

perturbation $||\tau||_2$ for every example in the test set. Afterward, we check how many of the calculated adversarial perturbations are smaller than $\eta\tau_0$ for different values of η. The results can be viewed in Figure 3.6.

The second experiment tests a more complex scenario. Here, data is situated on a smaller dimensional data manifold in the input space. To control for this, we compare the original dataset to a dataset that is randomly linearly transformed and embedded in an input space of a higher dimension. None of these two operations changes the inner structure of the dataset. Consequently, none of these should change the adversarial vulnerability of a classifier trained on this dataset. The results can be found in Figure 3.6.

While the actual norm of the perturbation needed to manufacture an adversarial example is very different for the original, the rescaled and the transformed dataset, we observe that this effect is accounted for by τ^0, which is 1290 ± 55.33 for the original, 5.07 ± 0.32 for the rescaled and 387.90 ± 23.85 for the transformed dataset. The number of successful adversarial attacks, measured by one minus the accuracy on the adversarial examples, behaves similarly with respect to η. Hence, we conclude that η allows us to compare adversarial robustness across datasets better than the original parameter $||\tau||_2$.

3.3.2 Experiments on the Measure $\mathfrak{R}$ Based on Perceptibility

We describe two experiments to demonstrate that our proposed score $\mathfrak{R}$ can identify sets of adversarial examples. To this end, we show that it agrees with and quantifies two qualitative observations from the literature.

The first observation is made in Tanay and Griffin (2016). The authors compare two classifiers for a binary classification task. The task is to distinguish between two

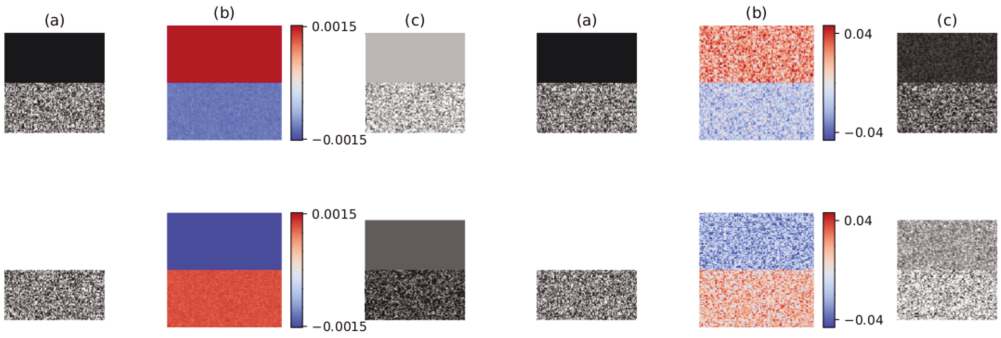


Figure 3.7: *For both cases of the first experiment, we show the original input in (a), the gradient of the loss function in of the classifier in (b) and the adversarial candidate in (c). The results on the left are generated by the classifier "with a proper amount of regularization" and the results on the right by the classifier "without regularization"*

Table 3.2: *Experimental results for the two examples from Tanay and Griffin (2016). Every experiment was run ten times, and we denote the result as mean ± standard deviation.*

Name	accuracy of F_O	accuracy of F_A	$\Re$
Box experiment ($\lambda = 0$)	1.0 ± 0	0.25 ± 0.25	-32.113 ± 11.166
Box experiment ($\lambda = 0.1$)	1.0 ± 0	1.0 ± 0	11.386 ± 0
Three vs Seven ($\lambda = 0$)	0.951 ± 0.000	0.264 ± 0.020	-4.280 ± 0.056
Three vs Seven ($\lambda = 0.1$)	0.951 ± 0.000	0.956 ± 0.010	1.038 ± 0.005

sets of 100×100 pixel images. The top halves of these images are black in one and white in the other set. In contrast, the bottom halves of images in both sets contains noise from a uniform distribution. The first classifier, which they call "without regularization," is a hard-margin support vector machine (SVM), and the second classifier, which they call "with a proper amount of regularization," is a soft-margin SVM. Instead of an SVM, we use a perceptron with and without L_2-regularization ($\lambda = 0.1$). The second experiment is identical, except the dataset is a subset of the MNIST dataset (LeCun, 1998) that contains only the images labeled as *three* or *seven*.

We calculate the adversarial candidates for both of these datasets using the DeepFool algorithm (Moosavi-Dezfooli et al., 2016).

The qualitative result of the first experiment is displayed in Figure 3.7. We observe a clear difference in the qualitative results of the classifiers. For the non-regularized classifier, the adversarial examples created from a clean example with dark upper half still has a dark upper half but are labeled as light upper half. The adversarial candidate does not resemble the target class with a light upper half. Hence, we can rightfully call these examples adversarial. For the regularized classifier, the adversarial example created from the example with a dark upper half has a top half closer to white than black (average value is bigger than 0.5). Thus, it resembles

the target class and does not fulfill the definition for an adversarial example. The results of the second experiment are similar. The adversarial candidates for the classifier regularized classifier visually resemble the target class, while for the classifier "without regularization" the adversarial candidates do not resemble the target class. Consequently, only the adversarial candidates for latter are adversarial examples.

In conclusion, the adversarial set contains real adversarial examples only in the case "without regularization." These results agree with the results reported by Tanay and Griffin (2016).

Our score $\mathfrak{R}$ in Table 3.2 quantifies these qualitative results. Our score detects in both cases correctly whether the adversarial candidates are adversarial examples or not. We run every experiment ten times. The score is ten times positive for the non-adversarial case and ten times negative for the adversarial case.

We conclude that our score $\mathfrak{R}$ can differentiate between examples that are visually similar to the target class and those that are not. The existence of the former is neither surprising nor problematic since large enough perturbations will always be able to change the decision of a classifier while the latter should not exist. These examples reinforce our belief that $\mathfrak{R}$ allows us to distinguish between meaningful features picked up by a neural network and noise even when domain knowledge is needed otherwise.

3.3.3 The Connection Between Multimodality and Adversarial Vulnerability

We need to be able to link properties of the dataset to adversarial robustness to understand the fundamental learning behavior of deep neural networks through why adversarial examples exist. In the previous sections, we took a step towards making adversarial robustness more comparable across datasets. In this section, we demonstrate an approach that utilizes this progress.

To demonstrate the usefulness of the results presented in Section 3.2, we show that multimodality of the distribution of individual classes is linked to adversarial robustness. To investigate this link, we need datasets that are the same in every way except the modality of the distribution of examples of one class. To this end, we start with the MNIST dataset and create new datasets by grouping several digits into one class.

Since the images of one digit in the MNIST dataset are similar, we expect their distribution to be unimodal. For our new datasets, where we grouped k digits into one class, the different digits within one class are not visually similar. Hence, we expect the resulting distribution to be k-modal.

We partition the ten MNIST classes into $m \in \{2, \dots 10\}$ classes, such that the number of classes in each part is as equal as possible. This new data set is called MNIST(m). Note that MNIST(10) is equal to the original MNIST. We train a classifier F_O on each MNIST(m) dataset. We use these new labels for the MNIST set and the DeepFool algorithm to calculate the adversarial set described in Section 3.2.

Afterward, we train a second neural network F_A on the adversarial set and calculate $\mathfrak{R}$ to determine whether the DeepFool algorithm found real adversarial examples.

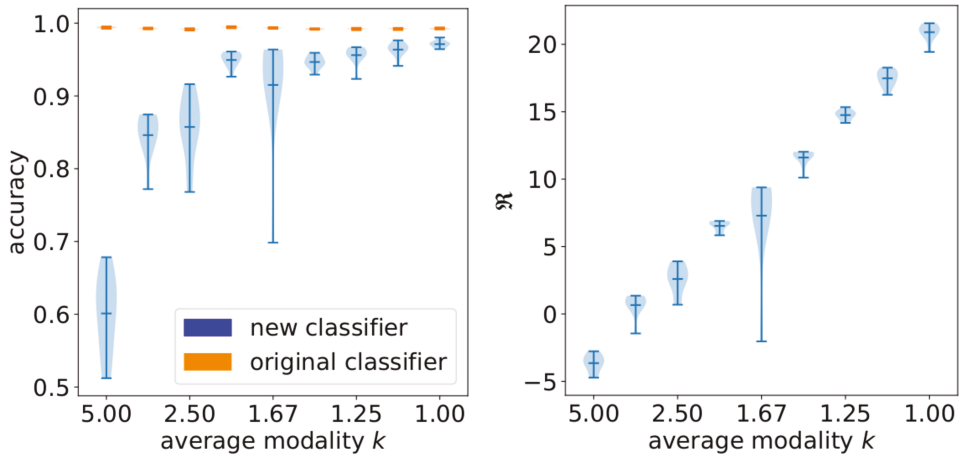


Figure 3.8: *On the left, we plot the accuracy of the original classifier F_O and the new classifier F_A against the average modality k of all classes. On the right, $\mathfrak{R}$ is plotted against k.*

We run this experiment ten times for every value of m. The results are displayed in Figure 3.8. The original classifier F_O achieves an accuracy above 0.99 for all values of k. In contrast, the new classifier F_A reaches accuracies as low as 0.6 if $k = 5$ and below 0.9 if $k \geq 3$. Furthermore, k, the modality of the distribution averaged over all classes, and $\mathfrak{R}$ are associated. The coefficient of determination between k^{-1} and $\mathfrak{R}$ is 0.9679.

This experiment shows how the approach demonstrated in this work can link a property of the dataset to adversarial robustness.

The results suggest that neural networks struggle with classes with multi-modal distributions. To argue that this correlation is not spurious, we now present a minimal example that can be observed in Figure 3.9. We train the same binary classifier in two situations. First, the distribution of both classes is unimodal. In this case, our representative training set contains the subset D_1 of examples of class c_1 (blue in Figure 3.9) and subset D_2 of examples of class c_2 (red in Figure 3.9). The distribution in both sets is unimodal. For the second case, we add the set E containing more examples of c_2 where E has a different distribution than D_2. For our experiment D_1, D_2 and E are normally distributed but have different means and variances.

As shown in Figure 3.9, the input-space contains areas classified as c_2 in the dataset without E and as c_1 in the dataset containing E. Hence, one of the estimations is erroneous.

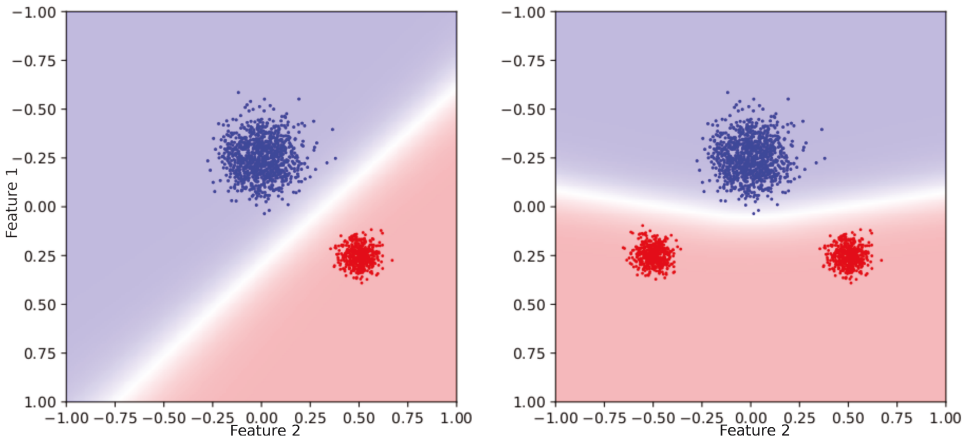


Figure 3.9: *In the left image, the decision boundary for the minimal example is plotted. In the upper plot, only D_1 and D_2 form the dataset. In the bottom plot, we added E. Some areas are classified as class c_2 in the upper plot and class c_1 in the right plot.*

4 | Determining the Relevance of Features for Deep Neural Networks

4.1 Theory

In this section, we introduce the new method we developed to determine which feature is used by a deep neural network. This method is based on the framework of causality introduced by Pearl (2009). We can formulate supervised learning as a structural causal model using this framework. Under some assumption on the supervised learning algorithm and some assumptions of causal inference, mentioned in Section 2.1, we can employ Reichenbach's common cause principle (Reichenbach, 1991) to reduce the question of whether a feature is relevant to the decision of a deep neural network to a conditional dependence test.

First, in Section 4.1.1, we motivate the basic idea of the method on example data from the national football league. Second, in Section 4.1.2, we thoroughly explain all variables and processes included in the structural causal model. To further illustrate these explanations, we provide an elementary example on synthetic data in Section 4.1.4. For the structural causal model described in Section 4.1.2 to be valid, we make some assumptions on the supervised learning algorithm. Hence, third, in Section 4.1.5, we investigate how to adapt the method demonstrated here if these assumptions do not hold. While the structural causal model is different if the assumptions do not hold, we end at the same conditional dependence test. Nevertheless, the method presented here has several shortcomings. We discuss these shortcommings in Section 4.1.6. Finally, in Section 4.1.7, we explain how our new method compares to state-of-the-art methods for understanding deep neural networks.

To demonstrate that this method is not limited to tasks on small synthetic datasets as presented in this section, we present multiple experiments in Section 4.3.1. We show that our method can be applied to complex real-world classifiers.

4.1.1 Motivational Example

We start by illustrating the method's main idea for understanding whether a feature is used by a supervised learning method. The core of the method is the conditional dependence test

$$X \perp\!\!\!\perp P \,|\, L \tag{4.1}$$

between the feature X and the prediction P of the supervised learning method given the true label L. We especially want to demonstrate that this is intuitively more reasonable than considering the unconditional dependence test

$$X \perp\!\!\!\perp P. \tag{4.2}$$

The first example we use is a two-class classification task, where three-dimensional inputs are classified as one out of $L \in \{0, 1\}$. The inputs are given by

$$I = \begin{pmatrix} i_1 \\ i_2 \end{pmatrix}, \quad I \sim \mathcal{N}\left(\begin{pmatrix} L \\ L \end{pmatrix}, \begin{pmatrix} 0.1 & 0 \\ 0 & 0.1 \end{pmatrix} \right). \tag{4.3}$$

Both dimensions on their own contain enough information to classify most examples correctly. This task corresponds to a situation where we want to classify inputs with multiple unique features. A classifier could extract any one of these features for classification. Hence, we consider two classifiers, F_1 and F_2. The former is a simple logistic regression on the first dimension of the input I, meaning that the probability of the input I coming from class one is given by

$$\mathbb{P}(L = 1 \,|\, I) = F_1(I) = \frac{1}{1 + \exp\left(-\left(i_1 - \frac{1}{2}\right)\right)}. \tag{4.4}$$

Similar, the latter classifier F_2 is a simple logistic regression on the last dimension of the input

$$\mathbb{P}(L = 1 \,|\, I) = F_2(I) = \frac{1}{1 + \exp\left(-\left(i_2 - \frac{1}{2}\right)\right)}. \tag{4.5}$$

For both of the classifiers, the chance that they misclassify one example out of a training set of 200 is smaller than 10^{-4}. The predictions of the two classifiers on an example set can be observed in Figure 4.1.

Testing the dependence between the feature X and the prediction P of the classifier is a naive approach to test whether a black-box classifier uses it. To evaluate this for the two classifiers mentioned above, we use the correlation with a shuffle test to find the distribution under the assumption of independence. The results are presented in Table 4.1. We see a strong dependence between the prediction of each classifier and both features. The chance of such a dependence appearing randomly in a sample of independent data is smaller than one in a thousand. This correlation between the prediction P and the feature not used by the classifier is due to the strong correlation of the features. Both features are high for samples from class 1, and both features are low for samples from class 0. In comparison, the conditional

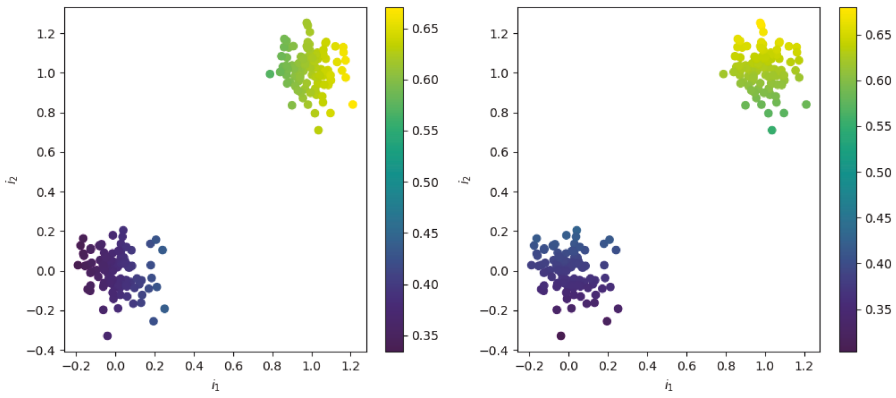


Figure 4.1: *A plot of 200 example inputs I for the synthetic example. The x-axis shows the first feature i_1, and the y-axis shows the second feature i_2. The color indicates the prediction of the classifier F_1 in the left plot and for classifier F_2 in the right plot.*

Table 4.1: *A comparison between the naive approach of calculating the dependence and our new approach to calculate the conditional dependence. The p-value gives the probability for a correlation equal or higher than the one observed under the assumption of independence. We reject the assumption of independence if this probability is below* 0.01

Classifier	True	$i_1 \not\perp\!\!\!\perp P$	$i_2 \not\perp\!\!\!\perp P$	$i_1 \not\perp\!\!\!\perp P \mid L$	$i_2 \not\perp\!\!\!\perp P \mid L$
F_1	i_1	✓ ($p < 0.001$)	✓ ($p < 0.001$)	✓ ($p < 0.001$)	✗ ($p = 0.987$)
F_2	i_2	✓ ($p < 0.001$)	✓ ($p < 0.001$)	✗ ($p = 0.987$)	✓ ($p < 0.001$)

dependence is only high between the prediction P and the feature used to calculate it. Obviously, the new conditional dependence is to be preferred over the naive approach of using unconditional dependence.

The second example shows that this observed advantage of conditional dependence over unconditional dependence is also visible on real data. To this end, we created a small dataset of American football players. In particular, we consider the wide receivers, tackles and centers from the Seattle Seahawks[1] and the New York Giants[2] rosters of the 2020/2021 Season. More specifically, we use two features of each player, namely the weight w in pounds and the acceleration measured by the time of a 40-yard-dash s in seconds taken from their respective Combine results[3]. The exact values can be found in Table 4.2. We use two simple logistic regression

[1] `https://www.seahawks.com/team/players-roster/`
[2] `https://www.giants.com/team/players-roster/`
[3] `https://nflcombineresults.com`

Table 4.2: *The real-world data used in the example. We report values for twenty players from the Seattle Seahawks (SS) and the New York Giants (NYG). We report their weight in pounds and the time they needed for a 40-yard-dash in seconds, both recorded at the NFL Combine. Finally, we report the position in which the players played in the 2019/2020 season as either tackle (T), center (C) or wide receiver (WR)*

Name	Team	Weight in lbs	40-Yard-Dash-Time in s	Position
Duane Brown	SS	315	5.08	T
Jamarco Jones	SS	299	5.50	T
Cedric Ogbuehi	SS	306	4.98	T
Brandon Shell	SS	324	5.22	T
Kyle Fuller	SS	320	5.24	C
Ethan Pocic	SS	320	5.15	C
Tyler Lockett	SS	182	4.44	WR
DK Metcalf	SS	235	4.33	WR
David Moore	SS	215	4.43	WR
Freddie Swain	SS	199	4.46	WR
Phillip Dorsett	SS	192	4.25	WR
Spencer Pulley	NYG	308	5.10	C
Jakson Barton	NYG	302	5.18	T
Cameron Fleming	NYG	323	5.28	T
Andrew Thomas	NYG	315	5.22	T
C.J. Board	NYG	181	4.42	WR
Austin Mack	NYG	215	4.59	WR
Sterling Shepard	NYG	201	4.48	WR
Darius Slayton	NYG	190	4.39	WR
Golden Tate	NYG	197	4.42	WR

classifiers similar to the synthetic data experiment above. The first classifier is

$$F_w = \frac{1}{1 + \exp\left(-\frac{w-\mu_w}{\sigma_w}\right)}, \tag{4.6}$$

where μ_w and σ_w are the expected value and the standard deviation of the weights. The second classifier is

$$F_s = \frac{1}{1 + \exp\left(-\frac{s-\mu_s}{\sigma_s}\right)}, \tag{4.7}$$

where μ_s and σ_s are the expected value and the standard deviation of the 40 yard dash times. A plot of the data can be found in Figure 4.2, and the calculated values are displayed in Table 4.3. We observe that the findings from the synthetic data extend to the real data.

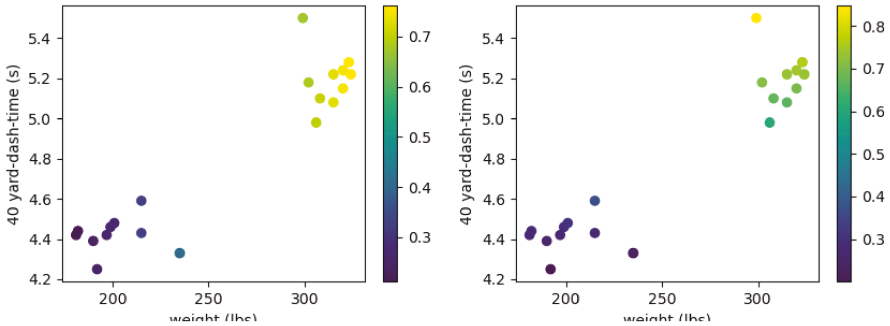


Figure 4.2: *The weight and the time needed for a 40 yard-dash for twenty professional American football players. The color in the left plot indicates the prediction of the classifier F_w, and the color in the right plot indicates the prediction of the classifier F_s.*

Table 4.3: *The results of the calculations in the American football example. The p-value gives the probability for a correlation equal or higher than the one observed under the assumption of independence. We reject the assumption of independence if this probability is below 0.01. We find that the unconditional tests reject the hypothesis of independence in every case, while the conditional dependence test rejects the hypothesis only in the correct cases*

Classifier	True	$w \not\perp\!\!\!\perp P$	$s \not\perp\!\!\!\perp P$	$w \not\perp\!\!\!\perp P \mid L$	$s \not\perp\!\!\!\perp P \mid L$
F_w	weight	✓ ($p < 0.001$)	✓ ($p < 0.001$)	✓ ($p < 0.001$)	✗ ($p = 0.903$)
F_s	40 yard dash	✓ ($p < 0.001$)	✓ ($p < 0.001$)	✗ ($p = 0.915$)	✓ ($p < 0.001$)

4.1.2 The Structural Causal Model: Variables and Functions

The task in supervised learning is to optimize a set of weights W of the learning algorithm. We use the inputs I and the corresponding labels Y from the training set TS to optimize these weights. These weights and the input image I are used to reach the prediction P of the automatic prediction method during the inference.

Hence, the three main Processes involved in supervised learning are the sampling process to sample examples from the underlying distribution, the training process used to optimize the weights and the inference function represented by the automatic prediction method after the training.

For the rest of this subsection, we will explain all of these variables and processes individually.

The Underlying Distribution of Inputs Parameterised by the Label L: As an example, consider a classification task in which we want to train a system to differentiate between images of cats and images of dogs. We need to acquire images from cats and dogs to train such a system. For the images of dogs, we start with a random dog

and then take an image under random conditions. The resulting image is labeled as "dog." Therefore, the underlying distribution of images in the classification task can be understood as the combination of the distribution of all possible images of dogs P_{dog} and the distribution of all images of cats P_{cats}, weighted by their respective priors $p(\text{dog})$ and $p(\text{cat})$

$$P = p(\text{dog})P_{\text{dog}} + p(\text{cat})P_{\text{cat}}. \tag{4.8}$$

Similarly, during inference, we take an image of a real animal. This real animal determines the correct label, and, hence, the image is not sampled from P but either P_{dog} or P_{cat}. In conclusion, when sampling for inference as well as when creating the training set, the images are sampled from the individual distributions P_{dog} and P_{cat} given the label and not directly from the more general distribution P. Consequently, we parametrize the distribution by the label and denote it by P_L. More generally, if we discriminate between n classes named $c_1 \dots c_n$, the distribution P is given by

$$P = \sum_{i=1}^{n} p(c_i)P_{c_i} \tag{4.9}$$

and when sampling, we sample from the distributions P_{c_i}.

The same holds if we are faced with a regression task. Instead of a class label, we have a prediction target Y, and the density P is given by the integral

$$P = \int_{\mathbb{R}} p(Y)P_Y dY. \tag{4.10}$$

If we sample an input example, we sample from the distributions P_Y.

As demonstrated in the motivational example in Section 4.1.1, we condition on this variable. Therefore, we need to understand the metric space formed by these distributions. To this end, we use the metric induced by the labels. This metric is in the trivial metric on the label space for classification tasks. From the third Postulate of Niemann (Niemann, 1990), we know that the domains occupied by features of different classes are different. Hence, the distributions for two classes, $L_1 \neq L_2$ are different from each other $P_{L_1} \neq P_{L_2}$. Therefore, we use the metric

$$d(P_{L_1}, P_{L_2}) = \begin{cases} 0 & P_{L_1} = P_{L_2} \\ 1 & P_{L_1} \neq P_{L_2} \end{cases} = \begin{cases} 0 & L_1 = L_2 \\ 1 & L_1 \neq L_2 \end{cases} \tag{4.11}$$

for a classification task.

The Training set TS: Since we want to learn the task at hand, we need a training set. A training set is generated by drawing Labels L and then drawing images from the respective distributions P_L. Following Niemann's postulates, we assume the training set represents the underlying distribution, meaning that the samples in the dataset are i.i.d. sampled from P.

The Weights of the Learning Algorithm W: A data-driven classification approach aims to find a function F that maps every input I onto the correct Label L

$$F : I \mapsto L. \tag{4.12}$$

Since we only have a limited amount of information, it is reasonable only to consider a particular class of functions. We parametrized this class of functions by a set of weights W. Consequently, this parametrization and the set of weights W are very different between algorithms. Here, we give some examples.

In deep neural networks, the weights on the connections between the neurons form the weights W. Similar, for a linear regression task, the weights W is the vector of the linear coefficient of the algorithm. In contrast, for example, in the case of the nearest neighbor classifier, the set of weights is the set of training images that are simply stored as weights for the inference process.

The Prediction P of the Automatic Prediction System: Even though we try to find a function F, which maps every input I onto the correct label L,

$$F : I \mapsto L, \tag{4.13}$$

we will not find such a function in most cases but will only find an approximation. One reason for this is that the inputs might be ambiguous, such that identical inputs might be labeled differently and, hence, we can only predict the most likely label. A second reason might be due to finite data. The effect of finite data might lead to some error in estimating the weights W. For this reason, the actual function we will find will not map to L but will map to an approximation

$$P = L + \epsilon, \tag{4.14}$$

where we model ϵ as random independent noise. We call this approximation of the label generated by the automatic system the prediction P of the algorithm.

The input Image represented as X and $\bar{X}$: One of the most important Variables in understanding whether a deep neural network uses a feature is is the input example. Critically, we have to separate the information in the input into the feature X we want to investigate and the information $\bar{X}$ orthogonal to this information. In the following, we will call X the feature of interest.

For example, if the input consists of multiple variables and the feature of interest is their mean, we can divide the information by using the mean as the feature of interest and the differences between each variable and this mean forme the set $\bar{X}$ of orthogonal features.

In this example, the feature of interest X is calculated by a mathematical formula. Besides this, there are at least two other ways to receive the feature of interest. The first is to use hand-annotation to get the value of X, and the second is to use a simpler learning algorithm to predict the feature of interest X for every input I. All

three of these methods of obtaining the feature of interest have advantages and disadvantages.

As for the idea of calculating the feature from the image analytically: The main reason we use a learning algorithm in the first place is that most semantically meaningful features are very hard to calculate using a mathematical formula. Hence, it might not be a valid approach for most interesting features to calculate it from the image deterministically. However, if we can calculate the feature of interest from the image directly, it is as close to the information presented to the algorithm. Both other methods will add some amount of noise to the feature of interest.

In comparison, relying on hand annotations is often the only reliable way to find the value for semantic features in an image. However, hand annotations require a lot of time and often expert knowledge. Hence, it is often not feasible to create annotations for images. Another drawback of this method is that an expert may use information outside of the image. Hence, the hand annotation might add some information not contained in the image, which means that the annotation is a noisy version of the information contained in the image.

Finally, we can use a combination of both methods. To this end, we need annotations on the feature of interest for some images, not necessarily images from the original data set. The main idea is to train an auxiliary simple supervised learning algorithm on these annotated images. Using this auxiliary algorithm, we get a prediction of the feature for every input in the training set. This auxiliary algorithm does not rely on additional information since it gets only the image as an input. However, the most important advantage compared to relying solely on annotations is that we need fewer annotations and that we can rely on other data sets that are annotated already. However, a different kind of noise is introduced. This noise originates from the approximation error of the auxiliary learning algorithm. Since we either rely on very few labeled examples or have to rely on examples from a slightly different domain, the simple classifier will not perfectly generalize to the original data set.

To illustrate the above possibilities, we present a simple example. Consider an application that identifies bird species from images. In particular, a supervised classifier is trained that distinguishes American crows (Corvus brachyrhynchos) from fish crows (Corvus ossifragus). Since these two bird species look very similar (The Cornell Lab of Orniothology, 2020), we might be worried that the classifier might use water in the background of images to indicate fish crows. If we employ the method described in this work to investigate this question, we need to extract a numerical feature that indicates whether and how much water is in the image;s background.

To this end, we have multiple options to evaluate the feature. First, we can use a mathematical formula. An example could be the cosine similarity between the input image I and a blue image B of the same size

$$X_1 = \frac{\langle I, B \rangle}{||I||\,||B||}, \tag{4.15}$$

the cosine similarity between the bottom half of the input image $I_{1/2}$ and a blue image $B_{1/2}$ of appropriate size

$$X_2 = \frac{\langle I_{1/2}, B_{1/2} \rangle}{||I_{1/2}||\,||B_{1/2}||} \tag{4.16}$$

or something similar.

Second, we can use hand annotations. The person taking the images must also note whether water is in the area the image is taken. If no such labels are provided, we must manually label every image in the dataset. These labels could be binary, one if there is water in the image and zero otherwise, or could, for example, denote the number of pixels that contain water.

Third, we can either manually label some of the images, for example, ten percent, of the dataset as described above or use images that contain no birds, but of which we know whether they contain water. We then use these images to train the auxiliary classifier, which could be, for example, a simple support vector machine, logistic regression or a small neural network.

The sampling processes S_T and S_F: The first set of processes involved in supervised learning are the sampling processes. The first sampling process, S_T, is used to sample the training set, and the sampling process, S_F, is used to sample examples during inference.

To sample the training set, we start by sampling a sequence of labels according to our prior distribution p_{l}

$$S_{T1}(\omega, P_{\text{prior}}) = (l_1, \ldots, l_m). \tag{4.17}$$

Given this sequence of labels, we use a second sampling function S_2 to sample an example from the distributions of images corresponding to the label

$$\begin{aligned} S_{T2}(\omega, (l_1, \ldots l_m), \{P_l\}) &= ((l_1, I_1), \ldots, (l_m, I_m)) \\ &= ((l_1, \{X_1, \bar{X_1}\}), \ldots, (l_m, \{X_m, \bar{X_m}\})) \end{aligned} \tag{4.18}$$

where I_i is sampled from P_{l_i}. Hence, the sampling process S_T is the combination of the two abovementioned sampling processes

$$\begin{aligned} TS = S_T(\omega, \{P_l\}, P_{\text{prior}}) &= S_{T2}\left(\omega, S_{T1}(\omega, P_{\text{prior}}), \{P_l\}\right) \\ &= ((l_1, \{X_1, \bar{X_1}\}), \ldots, (l_m, \{X_m, \bar{X_m}\})). \end{aligned} \tag{4.19}$$

The second sampling process is S_F, used to sample single examples for inference. The process is very similar to the abovementioned process. It also consists of two subprocesses. The first subprocess samples a label l from P_{prior}. The second samples an image I from the corresponding distribution P_l. In contrast to S_T, this sampling process, however, samples only a single example instead of a set of examples. Hence,

the first sampling subprocess is given by

$$S_{F1}(\omega, P_{\text{prior}}) = l, \tag{4.20}$$

and the second subprocess is given by

$$S_{F2}(\omega, l, \{P_l\}) = I = \{X\} \cup \bar{X}. \tag{4.21}$$

The sampling process S_F is then given by combining these two subprocesses

$$S_F(\omega, P_{\text{prior}}, \{P_l\}, P_{\text{prior}}) = S_{F2}(\omega, S_{F1}(\omega, P_{\text{prior}}), \{P_l\}) = I = \{X\} \cup \bar{X}. \tag{4.22}$$

The Training Process T: The most apparent process in supervised learning is the training process T. The training process is used to find an optimal set of parameters for the inference function of the supervised learning algorithm. While some supervised learning algorithms are deterministic, many also include randomness in the training process. We illustrate both of these possibilities later in this section using an example. Since our main focus is on deep neural networks, we use a stochastic training function,

$$T(TS, \omega) = W. \tag{4.23}$$

We now proceed with the two examples. The first example for a deterministic training function is the nearest neighbor classifier. The training process simply stores the training set into the weights W for this classifier. Therefore, nearest neighbor classifiers have a deterministic training process. Another supervised learning method that uses a deterministic training function is a linear regression that minimizes the mean squared error, where we can find the optimal set of parameters analytically. For both of these situations, the training function is given by

$$T(TS) = W. \tag{4.24}$$

An example of a supervised learning algorithm that uses a stochastic training process is deep neural networks. The weights are usually initialized at random for deep neural networks and then updated iteratively using randomly selected minibatches. These details, the random initialization and the random selection of the mini-batches make the training process of neural networks a stochastic process

$$T(TS, \omega) = W. \tag{4.25}$$

Typically, supervised learning methods that use iterative optimization methods are stochastic, while methods that reach their set of weights analytically employ a deterministic training function.

The Inference Function F: The last process we consider is the inference process of the supervised learning algorithm. This process is the only one that can form

a causal connection between the feature of interest X and the prediction of the supervised learning algorithm P.

The process is very dependent on the specific supervised learning method. For example, for the nearest neighbor classifier, the inference function aggregates the label of the k nearest neighbors into one prediction. In the example of the linear classifier, the input is multiplied by the learned weights to get the prediction. In the example of a neural network, the weights are used to calculate the prediction via a forward pass of the classifier.

In all of these examples, we observe that the inference function uses the weights W of the supervised learning algorithm and at least some of the features from the set $\bar{X}$. We are left with two possible inference functions. The first, in which the supervised learning algorithm uses the feature of interest to calculate its prediction,

$$P = F(W, I) = F(W, \{X\} \cup \bar{X}), \tag{4.26}$$

and the second, in which it does not,

$$P = F(W, I) = F(W, \bar{X}). \tag{4.27}$$

4.1.3 Combining the Variables and Processes into a Structural Causal Model

We have introduced all relevant Variables and Processes involved in supervised learning in Section 4.1.2 and now combine them into a structural causal model. The intrinsic variables of the structural causal model are the distributions P_L, the training set TS, the weights of the supervised learning algorithm W, the feature of interest X, the set of orthogonal features $\bar{X}$ and the prediction of the supervised learning algorithm P. As described above, these variables arrange in the following structural causal model

$$TS = S_T(P_L) \tag{4.28}$$
$$W = T(TS) \tag{4.29}$$
$$\{X, \bar{X}\} = S_F(P_L) \tag{4.30}$$
$$P = F(W, X, \bar{X}) \quad \text{or} \quad P = F(W, \bar{X}). \tag{4.31}$$

We excluded all dependencies on intrinsic or constant variables for this structural causal model. For example, the sampling processes have the random choice of labels as intrinsic variables. However, in the structural causal model, they only depend on the specific distribution P_L.

The corresponding graphical model can be found in Figure 4.3. This graphical model and the structural causal model in equations (4.28) to (4.31) show that the remaining question is whether the inference function F of the supervised learning algorithm uses the feature of interest X to calculate the prediction P. To this end, we test whether there is a statistical dependence between these two variables. However,

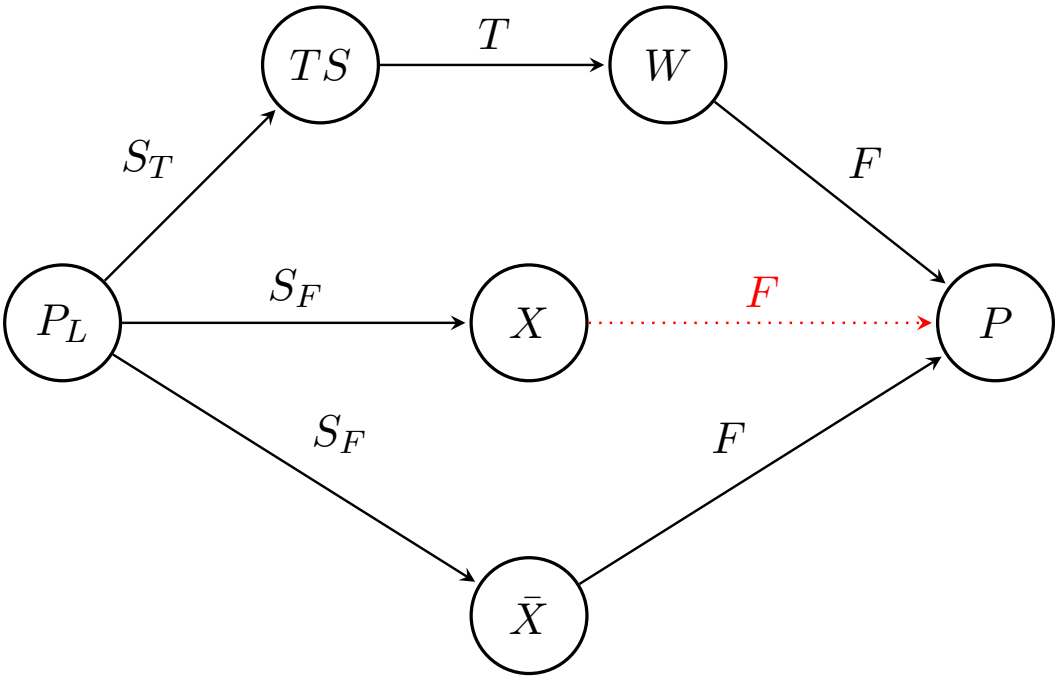


Figure 4.3: *The graphical model corresponding to the structural causal model displayed in equations (4.28) to (4.31). The intrinsic variables, namely the distribution of the examples of a specific label P_L, the training set TS, the weights of the supervised learning algorithm W, the feature of interest X, the set of features $\bar{X}$ that are orthogonal to the feature of interest and the prediction P of the supervised learning algorithm are displayed as the vertices of the graph. The arrows indicate the processes that connect these variables, namely the sampling processes S_T and S_F to sample the training set or single examples for inference, respectively, and the training process T and the inference process F. The remaining question is whether the inference function F uses the feature of interest X to calculate the prediction P. Therefore, the corresponding arrow is dashed and red.*

if we find dependence between these two variables, the direct connection between the variables X and P is not the only possible reason. Reichenbach's common cause principle states that if two variables X and P are dependent, then either one is causing the other, or there is a variable causing both,

$$X \not\perp\!\!\!\perp P \Rightarrow \begin{cases} X \to P & \text{or} \\ P \to X & \text{or} \\ \exists Z : X \leftarrow Z \to P. \end{cases} \tag{4.32}$$

We first investigate the third possibility. According to the graphical model, besides the direct path $X \to P$, there exist two indirect paths. The first path is $X \leftarrow P_L \to TS \to W \to P$ and the second $X \leftarrow P_L \to \bar{X} \to P$. The variable that blocks both paths is P_L. Hence, to ensure that the direct path in the structural causal model exists, we condition the dependence test on P_L.

As discussed above in Section 4.1.2, the metric in the label space induces the

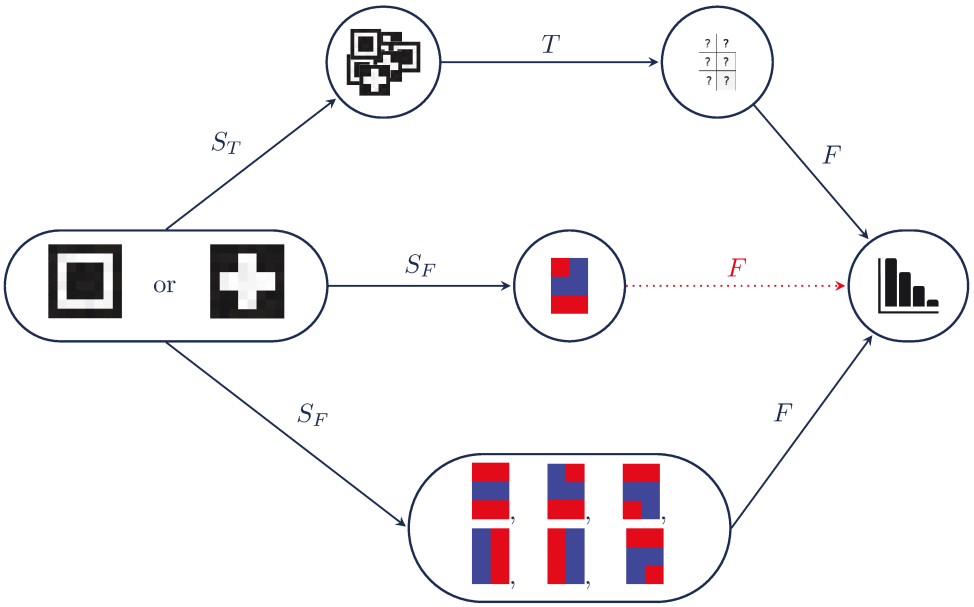


Figure 4.4: *The structural causal model for the illustrative example. The variables are the same as in Figure 4.3. Instead of the identifier, a visualization of the variable for the illustrative example is presented. The contents of this figure are similar to those of Figure 2 in Reimers et al. (2020). To emphasize the difference between input images and patterns representing features, we display the first in black and white and the latter in red and blue.*

metric in the space of the distributions P_L. Hence, conditioning on the distribution of examples P_L is the same as conditioning on the label L,

$$X \perp\!\!\!\perp P \,|\, P_L \Leftrightarrow X \perp\!\!\!\perp P \,|\, L. \tag{4.33}$$

If this conditional dependence test is still affirmative, then there is a direct link between the feature of interest X and the prediction P. The direction of this link is quite obvious. Since the value of the feature of interest can be evaluated independent of the supervised learning algorithm, the feature of interest X can not depend on the prediction P of this algorithm. Therefore, if the abovementioned test finds a conditional dependence between X and P, the only possibility is that the inference function F uses X as an input to predict P

$$X \not\perp\!\!\!\perp P \,|\, L \quad \Rightarrow \quad X \rightarrow P. \tag{4.34}$$

4.1.4 An Illustrative Example

To further illustrate the variables and processes and this new method to determine whether a feature is relevant, we apply the method to a synthetic example previously presented in Reimers et al. (2020).

Figure 4.5: *On the left and in the middle, two examples from the synthetic dataset are displayed. The high-intensity pixels form a square in the first image, and in the second image, the high-intensity pixels form a cross. On the right, we display all* 2×3 *patterns that, up to noise, appear in the images of squares but not in the images of crosses. To emphasize the difference between input images and patterns representing features, we display the first in black and white and the latter in red and blue. The content of this figure is taken from Figure 1 in Reimers et al. (2020).*

The synthetic dataset consists of mono-color images of 8×8 pixels. In each of these images, we find twenty pixels of high intensity, independently uniformly distributed over the interval $[0.9, 1]$, and 44 pixels of low intensity, independently uniformly distributed over the interval $[0, 0.1]$. The dataset is partitioned into two classes. The high-intensity pixels form a cross in one and a square in the other class. The task in this synthetic dataset is to distinguish between these two classes. We created 5000 examples from both classes forming a dataset of 10000 images. Examples of images for both classes are displayed in Figure 4.5.

Hence, the two distributions $P_{\square}$ and P_{+} for examples of the two classes are defined as

$$P_{\square} : x_{i,j} \sim \begin{cases} \mathcal{U}([0, 0.1]) & \text{pixel i,j dark in Figure 4.5 (left)} \\ \mathcal{U}([0.9, 1]) & \text{pixel i,j light in Figure 4.5 (left)} \end{cases} \tag{4.35}$$

and

$$P_{+} : x_{i,j} \sim \begin{cases} \mathcal{U}([0, 0.1]) & \text{pixel i,j dark in Figure 4.5 (middle)} \\ \mathcal{U}([0.9, 1]) & \text{pixel i,j light in Figure 4.5 (middle)} \end{cases} . \tag{4.36}$$

The sum of these distributions is the first variable involved in this example. The other variables are as follows. The training set TS is given by the 10000 examples sampled from these distributions.

We want a classifier that chooses one pattern and calculates the similarity between each image with this pattern. The highest resemblance to a patch of the image is the only relevant feature. We use a very simple convolutional neural network to build a classifier with this property. The network consists of one convolutional layer that uses only one filter of size 3×2, followed by a global max-pooling layer and a logistic regression for classification. It is reasonable to assume that the network

chooses a pattern that appears in every image of a square but no image of a cross. All seven patterns that appear in every image of a square but no image of a cross are displayed in the right of Figure 4.5. While the network can solve the task using another pattern, for example, a pattern that only appears in the cross images but not in the images of squares or patterns that do not use the full 3×2 patch but set some of the weights to zero, in our example it selected one of the patters in Figure 4.5.

Note that all patterns in the right of Figure 4.5 appear in every image of a square. No matter which pattern the classifier uses to detect the square, there will be a strong correlation between the appearance of the pattern and the classifier's prediction for each of these patterns, as reported in Table 4.4. Hence, the underlying distribution is a confounder for the existence of a pattern in the image and the classifier's prediction.

The classifier has seven weights. Six weights determine the pattern chosen by the convolutional layer, and one weight is used as a bias in this layer. For simplicity, in this example, we fix the bias to zero. Hence, the wheights W of the learning algorithm are given by the convolutional filter values.

We calculate these weights W for the convolutional neural network using the training algorithm T. In this case, we used stochastic gradient descent for ten epochs. The accuracy on a test set created the same way as the training set is 1.0, demonstrating that the learning algorithm identified a meaningful pattern.

One of the main reasons we use this simple classifier for this illustrative example is its interpretability. As explained above, the classifier selects one pattern and uses the highest similarity to a patch in the image for its decision. To understand the classifier, we can look at the learned weights of the kernel. In this example, the classifier used the horizontal stripe displayed in the first row of Table 4.4.

The function F we use in this example is given by the forward pass of the convolutional neural network.

The final variable is the feature of interest X. As an example, in the graph in Figure 4.4, we choose the highest cosine similarity between the bottom left corner shape and every 3×2 patch of the image. The set $\bar{X}$ contains all other features with no information on the cosine similarity between patches of the image and the bottom left corner shape. Even though we represent this set in Figure 4.4 by the other patterns, this set contains every orthorgonal feature, as discussed above. It is in no way limited to the similarity between the other patterns and patches of the input. At this point, we want to emphasize again that the method presented here is not suited to compare the use of different features. As mentioned, the seven patterns we investigate here are no complete list of possible patterns, and no such list is necessary. Every individual test for a feature is a test independent of other tests, and no comparison between these tests is reasonable.

To finish this illustrative example, we report the results of our method compared to the results using an unconditional correlation. We run the method three times, once for the horizontal stripe and once for each vertical stripe patterns displayed in Figure 4.5. We omitted the corner patterns from this investigation since they are,

Table 4.4: *Results for the illustrative example. The investigated feature is the maximum of the cosine similarity of the pattern and every patch of each image. The blue squares correspond to a value of one, and the red squares correspond to a value of minus one. For the unconditional test based on the coefficient of determination and the conditional test based on the conditional coefficient of determination, we report the test statistic, the p-value and whether the correlation is considered significant. Finally, we report whether the pattern was identical to the kernel of the neural network. For the kernel of the neural network, red denotes values below -1, and blue squares denote values above 0.9. Due to the confounding, the unconditional correlation evaluates all patterns as relevant, while our method only identifies the correct pattern.*

Pattern	$r^2(X,P)$			$X \not\perp\!\!\!\perp P \mid GT$			Equal to
	value	p-value	sig.	value	p-value	sig.	kernel: 
	0.997	0.000	**Yes**	0.693	0.000	**Yes**	**Yes**
	0.825	0.000	**Yes**	$1e^{-4}$	0.291	No	No
	0.827	0.000	**Yes**	$1e^{-4}$	0.411	No	No

through their computation, not otthogonal to the other patterns. We compare two tests: an unconditional correlation test and a partial correlation test conditioned on the ground truth label. We describe both these tests in Section 2.3.1. We reject the hypothesis of independence if the probability of the test statistic to be equal or greater than the observed value (p-value) is smaller than 0.05.

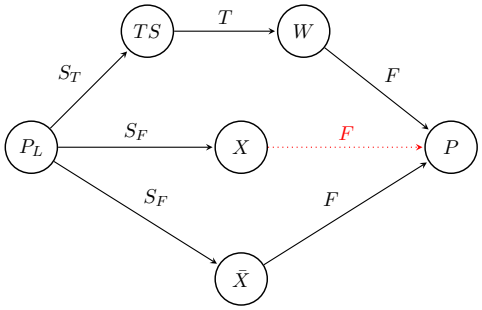
The results are displayed in Table 4.4. The table shows that the coefficient of determination is high for all three patterns. This observation is unsurprising since all patterns appear in every image of squares and no image of crosses. Hence, using the unconditional dependence, we would conclude that the neural network uses each of these patterns. However, conditioning on the ground truth eliminates the correlation between the false patterns and leaves a significant correlation only for the true kernel of the classifier.

4.1.5 Learning Algorithms which violate the Abovementioned Assumptions

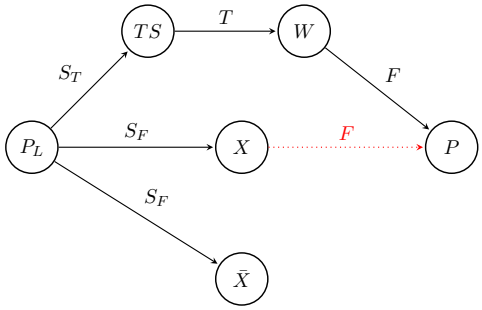
The structural causal model described in Section 4.1.2 relies on multiple assumptions. First, we assumed that the inference function of the classifier uses its weights W and at least one feature from the set $\bar{X}$. Furthermore, it relies on two assumptions from the causal inference framework of Pearl (2009), namely that the training and the inference function are stochastic. In this section, we will discuss how different supervised learning algorithms that do not fulfill these assumptions lead to different graphical models. However, we find that non of these changes influences the final result of

$$X \not\perp\!\!\!\perp P \mid L \quad \Rightarrow \quad X \to P. \tag{4.37}$$

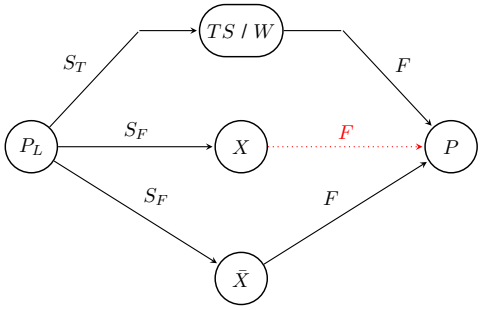
We start with the assumption that the inference process F uses its weights W



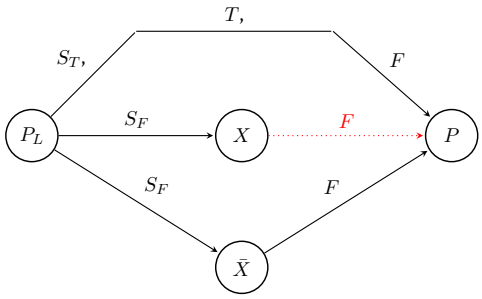
(a) *The original graphical model.*



(b) *The graphical model if the inference function does not use any feature from the set $\bar{X}$.*



(c) *The graphical model if the trainnig or inference process is deterministic.*



(d) *The graphical model if the training and inference process are deterministic.*

Figure 4.6: *The possible graphical models for situations which deviate the scenario described in Section 4.1.2. We display this graphical model in **(a)**. In **(b)**, the model for the situation in which the inference function uses no feature besides X, the feature of interest. In **(c)**, we display the model for a supervised learning algorithm that uses a deterministic training or inference process. In **(d)**, we display the graphical model for a supervised learning algorithm that uses a deterministic training and inference process.*

and at least one feature different from the feature of interest X in Section 4.1.5.1. Then we discuss the assumptions that either the training process T or the inference process F are deterministic or that both of these processes are deterministic in Section 4.1.5.2.

4.1.5.1 The Assumption that the Inference Process F uses W and at Least on Feature from the Set $\bar{X}$

In Section 4.1.2, we assumed that the inference process uses W, the weights of the supervised learning algorithm, and some at least one feature from the set $\bar{X}$, the set of all orthogonal features to the feature of interest X.

We first discuss the assumption that the inference function uses the weights W. If the inference function did not use any weights, it would, be independent of the training set and, hence, would no longer be a learning algorithm. However, in this work, we only focus on supervised learning algorithms.

The second assumption is that the inference uses at least one feature other than the feature of interest X. This assumption does not influence the final result. If we

drop this assumption, the inference process shown in equation (4.26) becomes

$$P = F(W, I) = F(W, X), \tag{4.38}$$

and the inference process in equation (4.27) becomes

$$P = F(W). \tag{4.39}$$

Using these equations in the structural causal model in equation (4.31) leads to the graphical model in Figure 4.6b. Again, dependence between the feature of interest X and the prediction P can be explained by either a direct link given by the function F in equation (4.38) or the indirect path $X \leftarrow P_L \rightarrow TS \rightarrow W \rightarrow P$. To block this path, we can condition on any of the three variables P_L, TS and W. Hence, if we drop the assumption that at least one feature different from the feature of interest is used by the inference process of the supervised learning algorithm, we still conclude

$$X \not\perp\!\!\!\perp P \,|\, L \quad \Rightarrow \quad X \rightarrow P. \tag{4.40}$$

4.1.5.2 Deterministic Processes for Training and Inference

The other assumption that leads to the graphical model in Figure 4.6a is that the training and inference processes are stochastic. Here we demonstrate that these assumptions are not vital towards the final result.

First, we discuss the possibility that the training process is deterministic. This is the case , for example, in the nearest neighbor classifier or a linear classifier optimized with mean squared error, but not in, for example, deep neural networks. If the training process is deterministic, the weights W of the supervised learning algorithm contain no information not already included in the training set. Hence, we can drop the weights W as a variable and instead consider the calculation of the weights as part of the inference function. Hence, we drop the deterministic function from the structural causal model. Under this assumption, the inference function described in equation (4.26) can be written as

$$P = F'(TS, I) = F(T(TS), \{X\} \cup \bar{X}) \tag{4.41}$$

and the function in equation (4.27) as

$$P = F'(TS, I) = F(T(TS), \bar{X}). \tag{4.42}$$

The resulting graphical model is presented in Figure 4.6c. We observe that, besides the direct path $X \rightarrow P$, two indirect paths, namely $X \leftarrow P_L \rightarrow TS \rightarrow P$ and $X \leftarrow P_L \rightarrow \bar{X} \rightarrow P$ exist. Both of these paths can be blocked by conditioning on the distribution of images of one class, which, as discussed above, is the same as conditioning on the ground truth label. We find

$$X \not\perp\!\!\!\perp P \,|\, L \quad \Rightarrow \quad X \rightarrow P. \tag{4.43}$$

The second possibility we discuss here is that the inference process is deterministic. In this case, as in the case where the training function is deterministic, we can interpret the inference function as taking the training set W as an input instead of the weights W. As in the previous example, the inference function described in equation (4.26) can be written as

$$P = F'(W, I) = F(W, \{X\} \cup \bar{X}) \tag{4.44}$$

and the function in equation (4.27) as

$$P = F'(W, I) = F(W, \bar{X}). \tag{4.45}$$

The resulting graphical model is presented in Figure 4.6c. In contrast to the previous example, this change is to include the stochasticity of the training process T into the inference process F. Since the graphical model is the same as in the example above, besides the direct path $X \to P$, there are the two indirect paths $X \leftarrow P_L \to W \to P$ and $X \leftarrow P_L \to \bar{X} \to P$. Both of these paths can be blocked by conditioning on the distribution of images of one class, which, as discussed above, is the same as conditioning on the ground truth label. We find

$$X \not\perp\!\!\!\perp P \,|\, L \quad \Rightarrow \quad X \to P. \tag{4.46}$$

The third possibility, in which the model can differ from the one displayed in Figure 4.6a, is that the training process T and the inference process F are both deterministic. In this case, we include the sampling process S_T, the training process T and the inference process F into one process. Hence, this new process is stochastic due to the stochasticity in the sampling process. Thus, the resulting structural causal model is given by

$$X, \bar{X} = S_F(P_L) \tag{4.47}$$

$$P = F(P_L, \{X\} \cup \bar{X}) \quad \text{or} \quad P = F(P_L, \bar{X}) \tag{4.48}$$

The corresponding graphical model is presented in Figure 4.6d. In addition to the direct path $X \to P$, two indirect paths, namely $X \leftarrow P_L \to P$ and $X \leftarrow P_L \to \bar{X} \to P$. Both of these paths can be blocked by conditioning on P_L. Hence, as in the above scenarios, we find

$$X \not\perp\!\!\!\perp P \,|\, L \quad \Rightarrow \quad X \to P. \tag{4.49}$$

These cases show that even if the assumptions we made to reach the structural causal model in equations (4.28) to (4.31) and the graphical model in Figure 4.3 are violated, the final result remains the same. Suppose we want to determine whether a supervised learning algorithm uses a feature of interest X. In that case, we can evaluate whether the feature of interest X and the prediction P are dependent given the label L.

We demonstrated that in the structural causal model, we can reduce the question of whether the supervised learning algorithm uses the feature X to calculate its

prediction P to a conditional dependence test $X \not\perp\!\!\!\perp P \mid L$. However, as discussed in Section 2.1, not every situation can be modeled using a structural causal model. In this section, we discuss why the structural causal model is a suitable model for this situation of supervised learning. Further, we will discuss the possibility of hidden confounders in the data generation process, the implications and possibilities we have to mitigate problems coming from hidden confounders.

4.1.5.3 The Possibility of Hidden Confounders

As described in Section 2.1, not every situation can be modeled using a structural causal model. The most critical assumption is that we can find a set of functions that form a directed acyclic graph. To this end, the situation of supervised learning has some advantages. The first two processes involved in supervised learning, namely the training process T and the inference process F, are mathematical algorithms. As such, they have a natural representation as a function. As such, they not only naturally form a directed acyclic graph, but we can further ensure that no hidden confounder between the weights W of the supervised learning algorithm and any other variable or between the prediction P of the supervised learning algorithm and any other variable.

Further, we consider possible causal relations between the feature of interest X and the set of all orthogonal features $\bar{X}$. We first emphasize that there is no direct causal link between them. To this end, we use the example of an image as input I. If the input is an image, the feature of interest will be some property of this image. For the set $\bar{X}$ of orthogonal features, we include all properties of the image that, through calculating them, do not depend on the feature of interest. To illustrate the connection between the feature of interest, we use the following example:

Consider an input image I. As the feature of interest, we take the red value of the top left corner

$$X = r_{0,0}. \tag{4.50}$$

One example feature that will be in the set $\bar{X}$ will then be the green and blue values of the same pixel

$$g_{0,0}, b_{0,0} \in \bar{X}. \tag{4.51}$$

While, due to the task we look at, there might be information on these values in the red value, they are not connected through the way they are calculated from the image. In contrast, for example, the sum of these color values is not in the set $\bar{X}$,

$$r_{0,0} + g_{0,0} + b_{0,0} \notin \bar{X} \tag{4.52}$$

because the feature of interest $r_{0,0} = X$ is directly involved in calculating this feature.

As is obvious from this example, there can not be a direct link $X \not\to \bar{X}$ from the feature of interest X to any feature from the set of orthogonal features $\bar{X}$. If not convinced, we can conduct a simple intervention experiment. If we change the feature of interest in an image, the features in the set of orthogonal features will not

change automatically. The same is true the other way around. Since the features in the set $\bar{X}$ do not contain any information on the feature of interest, there can also be no direct causal link $\bar{X} \not\to X$. Hence, no direct causal link between the two different features X, $\bar{X}$ exists.

One final causal relation could be missing from the structural causal model in equations (4.28) to (4.31) and the graphical model displayed in Figure 4.3. This link connects the training set TS and either X, the feature of interest, or $\bar{X}$, the set of features orthogonal to the feature of interest. However, since the training TS set and the images I are sampled independently, there can not be a direct causal link between them. Further, since the distribution of both is determined completely by the variable P_L and all additional information is completely random and independent between the training set TS and the features X and $\bar{X}$, there can further not be a hidden confounder between the training set TS and the example $I = \{X, \bar{X}\}$. The same is true for hidden confounders between the different features of the input image I. There is no hidden confounder between X and $\bar{X}$.

4.1.6 Limitations of Our Method

Despite the demonstrated independence to the assumptions leading to the exact structural causal model and the promising results of the two illustrative examples, it is not reasonable to apply the method in every situation. In this section, we discuss the limitations of the method presented here.

The first limitation is the topic of redundancy in features. At this point, it is important to emphasize this method is not built to compare different features. The existence of the set $\bar{X}$ of orthogonal features does not mean that any feature that is different from X, the feature of interest, is in this set.

To explain this further, we use the following example of two features: the volume of an object and the height of an object. To this end, consider a classification task of objects and a labeled training set. We want to understand if the supervised learning algorithm uses the height of the objects. We want to understand if the supervised learning algorithm uses the object's volume to calculate its prediction. To answer these two questions, we have to construct two different structural causal models. While most of the variables in the two structural causal models are the same, namely the distributions P_L, the training set TS, the weights W and the predictions P of the supervised learning algorithm. The other two variables are different for the two variables. Obviously, the feature of interest is the height in the first example and the volume in the second example

$$X_h = h \tag{4.53}$$

$$X_V = V. \tag{4.54}$$

Consequently, the sets $\bar{X_h}$ and $\bar{X_V}$ of orthogonal features are different between the two structural causal models. It is important that neither the set $\bar{X_h}$ contains the object's volume nor the set $\bar{X_V}$ the object's height. The reason for this observation

is that the two features are not orthogonal. This implies some important limitations of this method.

The first limitation is that the method can not differentiate between features that are bijective functions of each other. The reason for this limitation is that we rely on statistical dependence tests that detect whether two variables contain information on each other. If two variables are connected through a bijective function, they contain the same information. Hence, dependence tests can not tell these variables apart. Reimers et al. (2020) state that in a situation where the supervised learning algorithm ended at the function $f(x^3)$, no one can differentiate whether the function uses x^3 as an input or x. Further, such differentiation is neither possible nor desirable. Both options have to be accepted as correct, even though the values of these two features might seem to be non-trivially different. Another situation is if a feature contains more information than is needed. Hence, another feature relevant to calculating $f(x^3)$ is the feature (x, y). Since this feature contains the relevant feature x, it will be detetected as relevant. Reimers et al. (2020) present a further example to emphasize this further. This example is that any function $f(x)$ can be understood as the function $f(\log(\exp(x)))$. As in the example above, it is impossible to distinguish between the features x and $\exp(x)$ or any other feature connected through an invertible mapping to x.

Unfortunately, as highlighted in the example above, the semantically meaningful features in many real-world examples are not orthogonal. In the example above, we considered the volume and height of objects. An object of a specific height can have any volume, and we can vary the volume of an object without changing its height. The same is true if we exchange the height and volume of the feature. Hence, these two features might seem independent. However, in almost all datasets, these two features will be correlated. Hence, the features are not orthogonal and can not appear in the same structural causal model described above. This demonstrates a further limitation of our method. If we test whether the supervised learning algorithm uses the height of an object, it will answer affirmatively if the supervised learning algorithm actually uses the volume of the object because the height of the object contains information on the volume.

These challenges limit the use of our method. However, the following considerations can mitigate them and help users employ the method in many situations.

First, one might argue that in the case of height and volume, determining that both are used if one of them is used is the desired behavior, since naively increasing the height of an object will increase the object's volume and hence the prediction of the supervised learning algorithm. Further, if we want to investigate whether the height of an object influences the prediction of a supervised learning algorithm besides the influence of the volume, we can do so by choosing a more appropriate feature, for example,

$$X_{h \setminus v} = h - \mathbb{E}(h \mid v). \tag{4.55}$$

Note that this can only happen because the two features are related in every class, and the relation between the two is the same throughout the data.

Second, this problem of redundancy in features is common in low-dimensional

inputs. It is easy to see that most features will be related through the single variable, in the extreme case of a one-dimensional input. However, many independent and orthogonal features will appear in a high-dimensional space. We expect many false positives in simple low-dimensional examples, while we expect only little false positives due to this effect in high-dimensional real-world examples.

Another way to mitigate this problem can be taken from other explanation methods, such as, for example, explaining decisions by examples (Section 4.1.7.6) or feature visualization (Section 4.1.7.5). Here, instead of one, many similar explanations can be created, and a human observer has to find the essence of these explanations. Analogously, if we identify a group of relevant features, a human observer can look at these features and understand their essence.

Further, this method is not designed as an explorative method. Since every test is individual and independent of all other tests, it can not be used to rate the importance of a feature and identify a more important or less important feature. We envision the use of the methods in situations where a lot of prior knowledge exists, and users want to test whether a specific feature is relevant to the trained learning algorithm. This situation is common in medicine or climate science, where we have a lot of prior knowledge. It is further useful in situations of algorithmic fairness or debiasing in which we want to understand the influence of one specific, predefined feature.

As mentioned in Reimers et al. (2019), we rely on the assumptions of causal inference. These assumptions can be violated even in simple situations such as an XOR-gate, a trivial function or effects that cancel out. We discuss these assumptions in Section 2.1.

Finally, testing conditional independence on finite data samples is a challenging problem in itself. Additional assumptions often have to be made depending on the dependence test we apply. We discuss the dependence test we used in this work in Section 2.3. When using the method from this work, we recommend spending enough time to decide on an appropriate conditional independence test based on the data at hand.

4.1.7 Comparison to Existing Methods

Since understanding which feature is used by a deep neural network is an important challenge, different methods to evaluate it were proposed. We describe the methods in Section 2.4. In this section, we explain the difference between our method and these methods proposed in the literature. We start by discussing the similarities and differences to saliency maps, which we introduce in Section 2.4.1, in Section 4.1.7.1 and to explaining classifiers by examples from the training set, which we introduce in Section 2.4.2.5, in Section 4.1.7.6. Both of these approaches have a different aim than our method. Instead of globally, they explain the classifier locally, i.e., they explain single decisions of a classifier. Hence, these methods can be applied in combination with our method in that they can find explanations for multiple

examples. These can be used to generate a hypothesis on which feature is important, and our method can be used to test this hypothesis.

Similarly, the method of feature visualization, which we discuss in Section 4.1.7.5, can be used to generate hypotheses. While this method is a global explanation method, its output heavily relies on the initialization and hyperparameters of the explanation method. Hence, a group of explanations can be generated, and a hypothesis can be induced from these explanations. Our method can then be employed to test this hypothesis.

Finally, the methods that are most similar to ours are the idea of testing with concept activation vectors, the method of causal concept effect and other methods that employ causality to explain the classifier globally. We discuss these methods in Section 4.1.7.2, Section 4.1.7.3 and Section 4.1.7.4, respectively.

4.1.7.1 Saliency Maps

The most common methods used to understand deep neural networks are saliency maps, which we explained in Section 2.4.1. Our method differs in multiple central points from saliency maps.

Firstly, As described in Section 2.4.1, saliency maps depend either on the classifier's derivative or on the classifier's output on slightly altered input images. None of these two methods are obviously a measure of the relevance of a specific feature. In contrast, our method is based on the framework of causal inference. As explained in Section 2.1, the framework of causal inference was developed to answer questions of why a process reaches a specific result and which features influence the result of the process. Hence, the framework of causal inference is uniquely qualified to tackle the question of whether a deep neural network uses a feature. This fact is further corroborated by different facts that are described in Section 2.4.1. Firstly, different methods to find saliency maps lead to vastly different results. Secondly, randomly-initialized neural networks produce saliency maps that are visually and quantitatively similar to those produced by trained deep neural networks (Adebayo et al., 2018). Thirdly, saliency maps fail to attribute correctly when a constant vector shift is applied (Kindermans et al., 2019). Finally, similar to the prediction of a deep neural network, the explanation of a saliency map is vulnerable to adversarial examples (Ghorbani et al., 2019).

The second difference is in the amount of information on the weights and predictions of the deep neural network that we need to calculate an explanation. For the first kind of saliency maps, the ones based on the gradient, and the third kind of saliency maps, the ones built similar to the Taylor expansion, we need information on all weights of the deep neural network to calculate the gradient. Hence, both of these explanations are only feasible in the white-box setting. The second method to generate saliency maps, the method based on replacing single pixels of the input image, is not only feasible in a white-box setting, but still requires us to be able to query the deep neural network. In contrast, the method we present here can be applied in a black-box setting. We neither need any information on the weights of the deep neural network nor do we need to query the neural network for

any result outside the data set. All we need to calculate whether the deep neural network uses a specific feature are the feature for the input images, the correct labels for the input images and the predictions for the input images. In particular, we need no additional predictions on altered images.

Thirdly, the scope of our explanation method is different from the scope of saliency maps. As described in Section 2.4.1, the goal of saliency maps is to explain individual decisions of a deep neural network, while our approach aims to explain the inference function learned by the deep neural network as a whole. Given F, the inference function of the deep neural network, we want to determine whether this function takes the feature of interest as an input.

Fourthly, the range of features for which we can determine whether it is used by the deep neural network is different. As explained in Section 2.4.1, saliency maps calculate the importance of every pixel in the input image. Saliency maps can only highlight pixels or areas (sets of pixels) of the input image. Consequently, saliency maps can only be used to determine whether a feature is used that is directly part of the input image. Furthermore, in most cases, a semantic interpretation of the saliency map is needed to map it to a semantic feature. We illustrate this limitation of saliency maps using an example, where we distinguish images of two kinds of birds, first the white-crowned sparrow (Zonotrichia leucophrys) and, second, the white-throated sparrow (Zonotrichia albicollis). Birds of these species look very similar to each other. The main differences are the white throat patch, which gives the sparrow its name, and the yellow lores, on which we focus for this example. An example image for both of these birds can be found in Figure 4.7. Further, in the same figure, we display the saliency maps, based on the gradient of the classifier, for a simple classifier based on the cosine similarity between the image and a yellow image of the same size. From these saliency maps, we find that the region that contains the yellow lore is highlighted. The yellow lore is represented directly by a region of the image. However, other features, for example, the missing yellow lore, can not be highlighted by a saliency map. Further, even if we accept the fact that the yellow lore is important, the saliency map does not help us to determine, which aspect of the yellow lores is important. It could, for example, be the area of the lores, the similarity to a specific color, or both.

The combination of the two later differences demonstrates an important use-case of our method together with saliency maps. As discussed in Section 4.1.6, one limitation of our method is that we need to know the exact feature beforehand and can not use our method exploratively. Hence, one approach to combineing saliency maps and our method is to use a saliency map to explore which region is important and then identify possible features that can be represented by this area. In the above example, we can use saliency maps to identify that the region containing the yellow lores is important. Afterward, we can identify features represented by the highlighted area. In this example, this could be the distance to the dark yellow in a color space, the area of the lore, or the number of yellow pixels in the image. Now, we can use the method presented here, to determine for each of these features independently, whether it is used by the classifier.

Figure 4.7: *Images of two bird species from the CUB200 dataset (Welinder et al., 2010): the white-crowned sparrow (Zonotrichia leucophrys) on the left and the white-throated sparrow (Zonotrichia albicollis) on the right. Further, in the second row, we display the saliency based on the gradient of a simple classifier. In the right saliency map, the yellow lore of the white-throated sparrow is highlighted.*

4.1.7.2 Quantitative Testing with Concept Activation Vectors (TCAV)

Using this method, as described in Section 2.4.2.1, the importance S of a concept towards the classification of the class is given by the directional derivative of the second part of the classifier, wich we call F_2, in the direction of the class activation vector v_C^R,

$$S = \mathrm{D}_{v_C^R} F_2 = \left\langle \nabla F_2, v_C^R \right\rangle . \tag{4.56}$$

The method of quantitative testing with concept activation is in many aspects similar to the method we propose here. First, TACV, as well as our method, provides global explanations. Both of them consider the classifier as a whole and do not produce explanations for single examples. If the two methods report that a feature or concept is used by the classifier, it is used in general. Second, in Section 4.1.2, we explained that one can use a simple learning algorithm such as a linear SVM to receive values for the feature of interest. This is similar to the linear classifier that is used in TACV. Even further, the directional derivative is similar to a correlation. Specifically, if we assume that the concept C is linear in R,

$$C = aR, \tag{4.57}$$

and that the function F_2 is linear,

$$F_2(R) = bR, \tag{4.58}$$

then we find

$$v_C^R = \frac{a}{||a||} \tag{4.59}$$

and

$$\nabla F_2 = b. \tag{4.60}$$

Hence, the importance of the concept C is given by

$$S = \langle v_C^R, \nabla F_2 \rangle = \frac{\langle a, b \rangle}{||a||}, \tag{4.61}$$

which is very similar to the correlation, which measures statistical dependence in the linear case of the concept and the classifier's prediction. More specifically, the correlation is given by

$$\text{Corr}(C, F(I)) = \frac{\langle aR, bR \rangle}{\sqrt{||aR||\,||bR||}}. \tag{4.62}$$

Since the effect of the distribution of R should cancel out in general, the difference between the correlation and the saliency given by the concept activation vector is the same up to a factor. Importantly, if the correlation is zero if the concept and the prediction are independent, then TACV will also assign a saliency of zero to this concept.

Despite the similarity of the two methods, they vary in key aspects. First, TACV only works on concepts, features that partition all input images into two disjoint parts, while our method can be used for any feature, for which we can assign a numerical value to every input. Even though this is a straight-up advantage of our method, it is a minor improvement since we could simply expand the TCAV method to any kind of feature by replacing the SVM that classifies the concepts by a support vector regression (SVR).

More importantly, as described above, the method of TACV is similar to testing the correlation between the prediction and the concept. Our method differs in two main aspects from this idea. Instead of the correlation, we, first, use a more general dependence test. This more general test also considers non-linear dependencies. Second, we use the conditional dependence test conditioned on the different distributions P_L from which the inputs of each class are sampled. This conditioning of the dependence test is important to avoid false positives due to confounding. In fact, Goyal et al. (2019) demonstrate that the performance of TCAV drops significantly in situations with heavy confounding.

4.1.7.3 Explaining Classifiers with Causal Concept Effect (CaCE)

Similar to our method, CaCE is based on the framework of causal inference. As described in Section 2.1, the framework of causal inference was created to answer questions, such as whether a variable is influencing another. Hence, it is a natural choice to tackle the problem at hand. A second similarity to our method is the focus on confounding. However, CaCE uses intervention to deal with the problem of confounding, while we use conditioning in the graphical model. Intervention is the most straightforward and the most agreed-on method to evaluate whether a relationship is causal. Furthermore, the method does not rely on assumptions. In contrast, our method relies on multiple assumptions that are necessary to establish the connection between causal links and conditional dependence, as discussed in Section 2.1. However, CaCE needs a generative model. It replaces the difficult task of determining whether a feature is used by a deep neural network with the difficult task of creating a generative model that can model the feature of interest independently from other features. In contrast, our method reduces the same task down to one conditional independence test. Testing for dependence is a well-studied area of mathematical statistics and is considerably easier than the problem of creating a generative model.

Therefore, CaCE is preferable if one has access to a strong generative model. However, in situations with little data or very complex relations in the data our method is preferable.

4.1.7.4 Methods That Use the Network Structure as Causal Model

Methods following this approach connect individual neurons to semantic concepts.

In contrast, our method considers the deep neural network as one process and a representation of the input image in a suitable feature space. Further, all methods that extract the structural causal model from the architecture of the deep neural network need access to the architecture and intermediate results of the deep neural network. Hence, this method only works in the white-box setting. In comparison, our method can be applied in a black-box setting, where we do not have access to the architecture, weights and intermediate results and can not query the neural network on images outside of the dataset.

Another difference between our method and methods that use the network architecture as a structural causal model is that we also consider the training of the classifier, while other methods only consider the trained neural network.

4.1.7.5 Feature Visualization

As described in Section 2.4.2.4, feature visualization relies on multiple unreasonable assumptions about neural networks. The existence of prototypical images is questionable, and there is no one-to-one mapping between neurons and semantic concepts. In contrast, no such conceptual problems exist if we employ our method. Further, this method can only be applied in the white-box setting, while our method can be applied in a black-box setting.

4.1.7.6 Explanations by Example

Despite these advances, some drawbacks of this method remain. First, to use this method for an explanation, one has to store the whole training set in addition to the model. Given the size of modern deep learning datasets, this is a severe limitation. In contrast, for our method, any dataset can be used to evaluate the distribution of predictions and features. Second, translating the output of the explanation method to a semantic feature is non-trivial. The explanation can only tell us which images are similar to our test image. It does not tell us which image feature is responsible for this similarity. A human observer has to look at the image, and figure out what they have in common. In comparison, our method answers a very specific question for one specific feature and no semantic interpretation of the result is necessary.

A further difference between our method and these methods is that our method works in a black-box setting, while these methods require access to intermediate results of the deep neural network. An advantage of the methods that explain by example is that they are explorative, while the method we developed is not. This advantage and the abovementioned problem of interpreting the results use the combination of the two methods attractive. A user can query the explanation by examples method for similar images, create a hypothesis on which feature is important towards the decision of a deep neural network and use our method to confirm or reject the hypothesis.

4.2 Applications

4.2.1 Direct Application to Skin Lesion Classification

In this section, we demonstrate a relevant use-case of the method presented in this work. To this end, we present a study that investigates which features are used by two groups of state-of-the-art classifiers, that employ deep neural networks to the problem of automatic skin lesion classification. We determine whether the classifiers use specific features which we split into three groups. The first group of features should have no or very little information on the class of the skin lesion. Therefore, it is highly unlikely that the classifiers use any of these features. The reason we included these features in this study is to ensure that the method does not produce a high number of false positives. The fact we find that none of these features are used by the classifiers inspires trust in the positive results we find later.

The second group of features contains the four features named in the dermoscopic ABCD rule introduced by Nachbar et al. (1994). We find that all automatic classifiers use at least some of these features. This fact furthers the trust in these systems and that they will generalize to new unseen patients. However, we also observe that the color feature is not used, even though it is considered very important by practitioners. To further investigate this observation, we construct three aditional ways to score the color feature.

Third, we investigate, whether the classifiers are so-called CleverHans-classifiers

(Lapuschkin et al., 2019) that use shortcuts and rely on known bias features that are connected to specific diagnoses in the training data but are not medically relevant to them. Unfortunately, we find that all classifiers use at least one of these biased features. We conclude, that more work is still necessary to increase the quality and generalizability of these classifiers. The main part of the research presented here was originally published in Reimers et al. (2021b).

The rest of this section is structured as follows: In Section 4.2.1.1 and Section 4.2.1.2, we explain why the method presented in this work is a good fit for this problem, especially compared to other methods, for example, saliency maps (compare Section 2.4.1). Afterward, we will explain the classifiers we investigate for this study in Section 4.2.1.3, and the features and how we score them in Section 4.2.1.4.

The experimental results of this study are presented in Section 4.3.2. More introduction to the problem of identifying features relevant for skin lesion classification is given in Section 1.2.

4.2.1.1 Suitability of Our Methode for this Application

Medical applications like this one, provide a set of specific challenges and requirements for methods aiming to understand automatic classifiers. In this section, we name these specific challenges and discuss why they favor our method over, for example, saliency maps which we explain in Section 2.4.1.

First, understanding the classifier and being able to reliable explain it is very important in medical tasks. Since the patient puts their health and well-being at risk, they have to make the final decision. However, to make an informed decision the methods we employ to provide the basis for this decision have to be reliable and understandable. For both of these aspects, it is important to be able to state whether a feature is used to generate the prediction of an automatic system. Hence, in medical tasks, it is of particular importance that a method used for understanding a classifier is theoretically sound.

Our method is based on the framework of causal inference. As explained in Section 2.1, this framework was developed to answer precisely these kinds of questions. Mathematical as well as philosophical discussions exist, for example, Pearl (2009) or Reichenbach (1991). In comparison, most other methods are mainly based on empirical results. For the example of saliency maps, we discuss some of the resulting problems in Section 2.4.1. For example, some explanations are insensitive to the trained weights of the deep neural network, as demonstrated by Adebayo et al. (2018), some explanations change dramatically under transformations that should be irrelevant to the classifier, as demonstrated by Kindermans et al. (2019), and some explanations can be manipulated by the kind of adversarial examples described in Section 3, as described by Ghorbani et al. (2019).

Another set of methods that is based on the strong theoretical foundation of causal inference consists of methods that aim to perturb the input directly, for example, Goyal et al. (2019). However, to perturb inputs in a meaningful way is difficult. To change relevant features often requires the researcher to change large parts of the image. These changes are not unique. If the feature we want to change

is, for example, symmetry to at least one axis through the center of gravity, there are infinite possibilities to change the image such that it becomes symmetric for one axis. More specifically, we can select any axis and replace one half of the image by mirroring the other half. This method, however, might change many other features in addition to the symmetry and, hence, the different possibilities of altering the symmetry might result in vastly different results. To alter one feature more precisely, Goyal et al. (2019) suggest using a generative adversarial network. However, medical data is often highly complex and, since data is difficult and expensive to obtain, datasets are often relatively small. Hence, it is difficult to train a generative model and the error introduced from the small sample size in training the generative model will propagate into the final result of determining the relevance of features.

In contrast, our method does not need to train a generative model and still can single out specific features. Hence, it is a good fit for the domain of small and complex medical datasets.

Further, to ensure trust in a classifier in the medical domain, it is important to verify that it is robust. Changes in the imaging system or other changes in the environment that do not influence the skin lesion class should not influence the prediction of the system. Hence, we need global explanations that can generalize to unseen situations. Local explanations, on the other hand, only explain specific decisions in hindsight. Since our method can identify whether a feature is used by a classifier globally, it is suited to inspire trust in the predictions of automatic classification systems.

One drawback of our proposed method is that it is not explorative. To test whether a feature is used, we need to not only define the feature beforehand but we need to calculate a value for each image. This might be difficult in situations where the automatic classifier solves a problem that we are not able to solve. However, this is not the situation in medical tasks. In these tasks, we have large amounts of prior knowledge that tell us which features are important from a medical point of view. Therefore, the challenge in the medical field is to ensure that a classifier is based on these meaningful features and not based on bias variables rather than exploring which features are used. Our method is suitable for this kind of analysis.

Finally, other methods, for example, saliency maps, see Section 2.4.1, only highlight areas of the input images and leave the semantic interpretation of these areas to the user. However, in medical tasks, many of the relevant features are high-level descriptors of the image that can not be represented by areas of the input image. For example, the symmetry of an image for any axis through its center, can not be highlighted by any area of the image. In situations like these saliency maps are not very useful and our method is more suitable.

4.2.1.2 Related Work

To the best of our knowledge, this study is the first to explicitly determine whether the medical features named in the ABCD Rule are used by deep neural network classifiers for automatic skin lesion classification. Previous work has only focused on a more general investigation into which feature is used.

The authors of previous work have often used saliency maps. For example, Young et al. (2019) use saliency maps to determine which areas of the input image are relevant to deep neural networks. However, due to the drawbacks of saliency maps, described in Section 4.2.1.1, their investigation is limited to coarse analyses of medical relavant features and bias features that can easily be localized in the image.

Determining whether a specific bias is used by a deep neural network for automatic skin lesion classification is investigated more frequently. A list of known artifacts in dermoscopy images was presented in Mishra and Celebi (2016). One example is colorful patches next to the skin lesions that were introduced by the SONIC dataset (Scope et al., 2016). Since this study investigats nevi, the colorful patches only appear in images with nevi. Hence, this dataset introduces a bias in the ISIC archive[4]. This bias influences the prediction of deep neural network classifiers. Rieger et al. (2020) demonstrated that their classifier "looks" at these patches when classifying skin lesions, using saliency maps.

Bissoto et al. (2019) propose a different approach. They replace the center of images containing the skin lesion with black boxes. Surprisingly, the performance of the deep neural network classifiers they investigate is still high, even if 90% of the image is covered by a black box and the skin lesion itself is no longer visible. The authors conclude, that deep neural networks do not rely on the ABCD rule features but use bias features instead. In further investigations, Bissoto et al. (2020), shows that this effect of high accuracy on images with no skin lesion in them decreases if during training the background of images is replaced by a mean background. Further, they score some of the main biases and measure the correlation to specific classes.

Another approach is to evaluate the classifier on two test sets. One of these test sets contains a certain bias and the second does not contain the bias. The comparison of accuracy is then used to evaluate the importance of the bias. This approach was, for example, used by Muckatira (2020). The authors find that the accuracy of their classifier varies across different age groups and between male and female patients.

All these methods only can investigate whether the classifiers use the foreground or the background of the image and whether they use biases that are tied to specific locations in the image, such as colorful patches or drawn scales. In comparison to these methods, our method allows us to investigate the specific features of the ABCD rule individually. In addition to the localized biases, it can also be used to investigate biases that are hard to localize, such as, for example, the age or sex of the patient.

4.2.1.3 Classifiers

In this section, we introduce the two classifiers we investigate in this study. We start with the classifiers introduced by Perez et al. (2018). We chose this classifier

[4] `www.isic-archive.com`

since it won the best paper award at the ISIC Skin Analysis Workshop at the 21st International Conference on Medical Image Computing and Computer Assisted Intervention 2018[5]. We train multiple versions of this classifier. The second classifier is the deep ensemble classifier presented by Gessert et al. (2020) that won the ISIC challenge 2019[6].

The main idea behind the classifier introduced by Perez et al. (2018) is test time augmentation. In test time augmentation, instead of classifying a test image, we create some number m of copies of the test image. Each of these copies is then independently, randomly augmented. The augmented copies are classified individually by the backbone classifier. Then the individual classification results are aggregated into one final classification.

Following the description, in Perez et al. (2018) we pre-train the classifier on ImageNet (Russakovsky et al., 2015) and train it on the ISIC 2017 challenge dataset (Codella et al., 2018). For the augmentations, we use the augmentation scenario "J" as described by Perez et al. (2018).

We train models of this kind for two different tasks. First, we train models for the recognition of melanoma, and, second, we train models for the recognition of seborrheic keratosis. For both of these tasks, we train deep neural network backbone classifiers with three different architectures. The first architecture is the ResNet-152 introduced by He et al. (2015), the second is the Inception-v4 architecture introduced by Szegedy et al. (2016) and the third is the DenseNet-161 architecture introduced by Huang et al. (2018). For each of these architectures, we compare a classifier that creates 26 augmented copies of every test image and a classifier that creates 52 augmented copies of every test image. Finally, we compare two different ways to aggregate the individual classification results. The first aggregation method is the mean of all classification scores and the second is to take the maximum over all the classification scores. This leaves us with 24 classifiers of this type.

The performance of the classifiers we trained can be found in Table 4.5. The performance we observe is similar to Perez et al. (2018) on the melanoma detection problem. For the seborrheic keratosis detection problem our models perform even better.

Table 4.5: *Performance of the Perez et al. (2018) models on the test set of the ISIC 2017 challenge (Codella et al., 2018). We report the area under receiver operating characteristic curve (AUC) and the accuracy (ACC). These results where originally published in Reimers et al. (2021b)*

	Melanoma		Seborr. Keratosis	
Backbone	AUC	ACC	AUC	ACC
ResNet-152 (He et al., 2015)	0.886	0.868	0.938	0.918
Inception-v4 (Szegedy et al., 2016)	0.851	0.835	0.924	0.887
DenseNet-161 (Huang et al., 2018)	0.877	0.852	0.944	0.913

[5] `https://workshop2018.isic-archive.com/`

[6] `https://challenge.isic-archive.com/leaderboards/2019`

The classifier proposed by Gessert et al. (2020) won the first two tasks of the ISIC 2019 Skin Lesion Classification Challenge. Besides the test time augmentation described above, the main technique employed by this classifier is to use an ensemble of deep neural network classifiers instead of a single deep neural network. To this end, Gessert et al. (2020) use mostly multi-resolution EfficientNets (Tan and Le, 2020) pre-trained on ImageNet (Russakovsky et al., 2015). To replicate the ensemble, we trained five EfficientNets (B0). Even though, this is a much smaller ensemble of classifiers it is the default parameter setting of the code provided by the authors.[7] Hence, although we expect this ensemble to work worse, we still expect it to be a reasonable representation of the original classifier.

We follow the training scheme proposed by Gessert et al. (2020). The classifiers are trained on three datasets, namely HAM10000 (Tschandl et al., 2018), BCN20000 (Combalia et al., 2019) and MSK (Codella et al., 2018). For more details on the training details and hyperparameters, we refer the reader to Gessert et al. (2020). Since this classifier is trained to perform a multiclass classification, we use it as the nine individual detectors for each class.

The performance of our smaller ensemble classifier is presented in Table 4.6. It is noticeably worse than the larger ensemble used in Gessert et al. (2020). It achieved a training accuracy of only 0.541.

Table 4.6: *AUC scores from an ensemble of five EfficientNets (B0) for different classes. These results were originally published in Reimers et al. (2021b)*

Class	AUC score
Melanoma	0.802
Melanocytic Nevus	0.884
Basal Cell Carcinoma	0.895
Actinic Keratosis	0.889
Benign Keratosis	0.819
Dermatofibroma	0.844
Vascular Lesion	0.938
Squamous Cell Carcinoma	0.882

These two classifiers, the one by Perez et al. (2018) and the one by Gessert et al. (2020), are a good representation of the state-of-the-art in automatic skin lesion classification. They contain the main ideas of ensembles of large, pre-trained deep neural networks and the idea of test time augmentation. They also cover melanoma detection as well as multiclass skin lesion classification.

4.2.1.4 Features

In our investigation, we use three groups of features.

First, we extracted four features from the input images that should contain none or very little information on the skin lesion in the image. We use these features

[7] `https://github.com/ngessert/isic2019`

to make sure that the method proposed here does not produce an unreasonable amount of false positives and, therefore, can be applied to this complex real-world problem of automatic skin lesion classification. We selected features for this group that match the meaningful and bias features in complexity and are derived similarly. The fact that our method correctly identifies that these features are extremely rarely used by any of the classifiers, reinsures us that a feature is used by a classifier if the method indicates it. The features we use for this purpose are:

- **Orientation:** The orientation feature is calculated by estimating the ellipse with the same second moments as the skin lesion's contour. We then measure the angle between the first axis of the image and the major axis of the ellipse. Since this feature purely includes information on the specific image and not on the skin lesion itself it should not be used by any classifier.

- **Random Symmetry:** This feature describes the symmetry of the contour of the skin lesion to a random axis that leads through its center of gravity. To evaluate the symmetry, we calculate the intersection over union between the contour of the skin lesion and the contour flipped along the random axis. We repeat the process for the orthogonal axis and multiply both intersection over unions. Even though symmetry is an important feature, and a perfectly round skin lesion will be symmetric to every axis, the symmetry to a random axis should not contain much information and should not be used by a classifier.

- **Image ID:** As the third feature, we use the position of the image in the ISIC archive (ISIC-archive, 2021). This feature contains some information since images from the same source receive consecutive numbers in the archive. However, this information is not very useful for the classification task and, hence, we expect the classifiers to ignore it.

- **MNIST class:** We train a classifier for hand-written digits on the MNIST dataset introduced by LeCun et al. (1998). We show the architecture of this classifier in Table 4.7. It is trained for 50 epochs using stochastic gradient descent with a learning rate of 0.01, Polyak-momentum (Polyak, 1964) of 0.9, and a weight decay of 0.0005. It reaches an accuracy of 0.9939 on the MNIST test set. The feature is then given by the prediction of this network for the segmentation mask of the skin lesion. Since this feature includes no useful information we expect all classifiers to ignore it.

We evaluate the distributions of all of these features on the HAM10000 dataset (Tschandl et al., 2018), which is part of the ISIC archive. It contains 10015 images of seven different classes. In addition, it contains the ground truth segmentations and different metadata such as the age and sex of patients.

The second group of features we investigate are features that are determined to be relevant by dermatologists. To this end, we consider the features named in the ABCD rule (Nachbar et al., 1994), namely the **A**symmetry, the **B**order, the **C**olor and

Table 4.7: *The architecture of our MNIST classifier*

Layertype	Filter Size	Filters	Padding
Convolutional	5×5	32	2×2
Convolutional	3×3	32	1×1
Max Pooling	2×2	-	2×2
Convolutional	3×3	64	1×1
Convolutional	3×3	64	1×1
Convolutional	3×3	128	1×1
Convolutional	3×3	10	1×1
Global Average Pooling	-	-	-

the **D**ermoscopic features. This rule was developed by dermatologists to identify melanoma. The exact way we score these features is as follows:

- **Asymmetry:** To score this feature, practitioners identify axes to which the skin lesion is almost symmetric. They find the maximum number of such axes through the center of gravity of the skin lesion that are orthogonal. In a two dimensional image, we can find a maximum of two such axes. Hence, this feature is zero, one, or two. To score the asymmetry automatically, we evaluate the intersection over union between the contour of the skin lesion and the contour of the same skin lesion fliped along an axis. We evaluate this value for 360 equidistand axis through the center of gravity of the skin lesion's contour. We say that the lesion is symmetric with respect to an axis if this intersection over union is larger than 0.9. Note that some researcher consider not just the contour but also the color of the skin lesion to score its symmetry. We decided against the use of color to generate a simpler feature and to distinguish it clearly from the color feature exaplained later. To find the contour of the lesion, we use the ground truth annotation for the semantic segmentation of the skin lesions provided by te HAM10000 dataset.

- **Border:** The goal of the border feature is to score whether the border of the skin lesion is sharp and regular. In the original description of the ABCD rule by Nachbar et al. (1994), this feature was scored by dividing the lesion into eight pieces and determining for how many of these the border is sharp and regular. This leads to a score between zero and eight. We use a different way to score this featur where we focus on the irregularity of the border. We consider the isoperimetric fraction, the fraction of the area A of the skin lesion and the squared length of its perimeter P,

$$\text{border} = \frac{4\pi A}{P^2}. \tag{4.63}$$

  This fraction should be small for lesions with irregular borders and is minimal for circular lesions. We evaluate this feature by using the handcrafted segmentation maps of the HAM10000 dataset.

Table 4.8: *Ranges in the HSV-colorspace corresponding to the individual colors named in the Color score of the ABCD-rule*

Color	Hue	Saturation	Value
White	0° - 360°	0% - 10%	90% - 100%
Red	350° - 25°	70% - 100%	80% - 100%
Light Brown	0° - 40°	50% - 100%	70% - 90%
Dark Brown	11° - 47°	60% - 100%	40% - 60%
Gray-Blue	216° - 252°	30% - 100%	60% - 100%
Black	0° - 360°	0% - 100%	0% - 15%

- **Color:** The next feature that dermatologists deem relevant is the color of the skin lesion. They consider the variability of color that appears in the skin lesion, more specifically, how many of six predefined colors, namely white, red, light brown, dark brown, blue-gray, and black appear in the skin lesion. This leads to a score between zero to six. The color in an image is not just determined by the color of an object, but also by, for example, the surrounding colors and the color of the light during the capturing of the image. Hence, interpreting the existence of a specific color as containing a pixel of the exact colors is too restrictive. To account for this fact, we assign a range of values in the HSV color space to each color. The color feature is then given by the number of color intervals for which the image contains a pixel with this color. The exact color intervals can be found in Table 4.8

- **Dermoscopic structures:** Dermoscopic structures are structures that appear in skin lesions. These structures are described in the Seven-Point Checklist proposed by Argenziano et al. (1998). The possible structures that are included in this feature are the milia like cysts, negative networks, pigment networks, streaks, and globules. We count how many of these structures appear in the skin lesion. To evaluate this feature, we rely on the ground truth annotation from the corresponding task in the 2018 ISIC challenge (Codella et al., 2018).

Since these features are relevant from a medical point of view, an optimal model should also use them to find the correct decision. To find the distribution of values for all of these features and the predictions we rely on the HAM10000 dataset (Tschandl et al., 2018) except for the last feature for which we use the 2018 ISIC challenge (Codella et al., 2018). These features have a similar complexity to the first four features that we use as a control group. For example, the asymmetry and border feature are similar to the random symmetry and the orientation feature above in complexity.

The experimental results in Section 4.3.2 indicate that the classifiers do not use the color feature, although it is considered one of the most important features by practitioners. To further investigate this fact, we calculated three additional ways to extract the color information that differs from the feature described in the ABCD

Table 4.9: *Relaxed ranges in the HSV-colorspace corresponding to the individual colors named in the Color score of the ABCD-rule*

Color	Hue	Saturation	Value
White	0° - 360°	0% - 10%	90% - 100%
Red	295° - 18°	40% - 100%	50% - 100%
Light Brown	349° - 43°	25% - 100%	60% - 90%
Dark Brown	349° - 43°	25% - 100%	10% - 60%
Gray-Blue	198° - 277°	30% - 100%	50% - 100%
Black	0° - 360°	0% - 100%	0% - 15%

rule but might have advantages in the automatic extraction and still capture the idea of quantifying the variability in color within the skin lesion. We compare their use to the color feature presented above, which we call "Color Count" throughout this experiment. The three features are:

- **Relaxed Color Count:** The color intervals that we selected for the color feature might be too restrictive. However, if we use intervals that are too large, the feature becomes meaningless. As a first alternative, we relax the intervals around the colors until it stops matching the described color. Table 4.9 contains the resulting interval borders, which are highly subjective.
- **Variance of the Hue:** Following Celebi et al. (2007), we use the variance of the color within the area of the lesion. While Celebi et al. (2007) suggest a wide variety of color channels, we decided on the hue in the HSV colorspace. The HSV colorspace separates the color from the saturation and luminosity. In comparison to the relaxed color count feature, this score is less subjective. However, it differs more from the feature described by Nachbar et al. (1994).
- **Volume in the RGB Color Space:** The color feature named in the ABCD rule counts the number of colors but does not consider the area that is colored this way. However, the last feature, i.e. , the variance, depends heavily on the area occupied by a specific color. As an alternative, we construct a point cloud in a three-dimensional space by converting each pixel in the skin lesion into one point using its RGB values as coordinates. We use the volume of the convex hull of this point cloud as the final color feature.

These features depict three common ideas when designing color features. The first feature determines relevant colors and measures whether they appear in the skin lesion by relaxing the intervals around them. The second feature is a more objective feature that does not contain a choice of colors but only the variance in color. The third also does not contain any color choices but is closer to the original feature. It does not consider the number of pixels of a specific color but only the variety of colors existing within the skin lesion. Furthermore, it also considers the luminosity and saturation of pixels.

The final, third, group of features are known bias features. First, features that are independent of the diagnosis in the real world, but are connected to some label in the dataset, for example, the colorful patches or the skin color of the patient. Second, features that we want the algorithm to ignore, even though they might influence the probability of a skin lesion being cancerous, for example, the age and sex of a patient. Independent of the differentiation into these two groups, we expect an optimal classifier to use none of these features. The four features we use in this group are the following:

- **Age:** The first feature which we want the classifier to ignore is the age of the patient. Since we expect the age of the patient to influence the appearance of the skin around the skin lesion, the classifier could extract this feature from the image. Even though the probability of melanoma increases with age, we want the algorithm to focus on the visual appearance of the skin lesion and leave the inclusion of other features like age to the practitioner. To score this feature, we rely on the annotations in the HAM10000 dataset (Tschandl et al., 2018). In this dataset, the age rounded to the next multiple of five of each patient is annotated.

- **Sex:** As a feature, the sex of the patient is very similar to the age of the patient. As for the age, it can be determined from the appearance of the skin around the skin lesion, it might be related to the risk to develop melanoma but we do not want the automatic classifier to use it. As for the age of the patient the the sex of the patient as a binary variable is annotated in the HAM10000 dataset (Tschandl et al., 2018).

- **Skin color:** The third feature is skin color. This feature should not be used by a classifier. To determine the skin color from an image, without considering the color of the skin lesion, we consider only the pixels in the top-left corner of the image. More specifically, we crop a ten-by-ten-pixel patch in the top-left corner of the image. Unfortunately, some images have a black border. To avoid problems from black borders, we excluded all images where the patch was black from further investigation. On the remaining images, we perform principal component analysis (PCA). We evaluate the score for this feature, we calculate the loading of the first principal component. In other words, if the top left corner patch $\mathbf{x}$ can be represented by the priciple components $\{\mathbf{e}_i\}_{i\in I}$ as

$$\mathbf{x} = \sum_{i=1}^{300} a_i \mathbf{e}_i, \tag{4.64}$$

  with a_i the loading of the i-th principal component. Then the feature is scored by a_1. Visual inspection of the results, which are presented in Figure 4.8, show that this loading corresponds to the lightness of the skin. A higher value is given to images of patients with darker skin.

- **Colorful patches:** The final feature we use in this study is the existence of colorful patches in the image. Through the inclusion of the SONIC dataset

Figure 4.8: *Examples of top-left ten-by-ten-pixel areas of images from the HAM1000 dataset ordered by the loading of the first principal component from lowest on the left, to highest on the right.*

(Scope et al., 2016) some of the images in the ISIC archive contain colorful patches, most of which also contain benign nevi. Figure 4.9 displays some examples. Deep neural networks can identify and utilize this relation between the colorful patches and benign lesions as demonstrated, for example, by Rieger et al. (2020). To score this feature, we count the number of pixels that display the colorful patch. To detect and localize these patches we use the patch segmentations provided by Rieger et al. (2020).

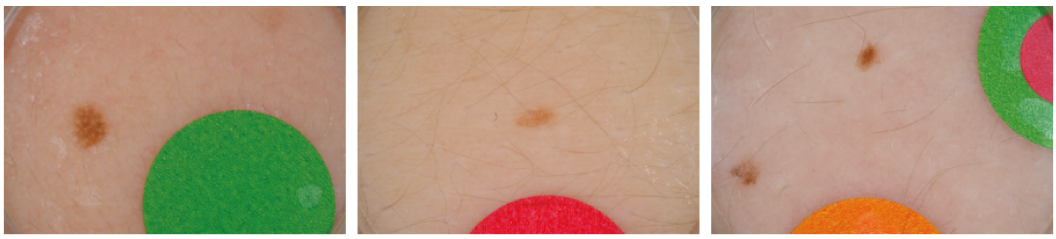


Figure 4.9: *Some images from the first 10K images of the ISIC archive. All images containing colorful patches and benign nevi in children from the SONIC dataset Scope et al. (2016).*

To evaluate the first three of these bias features we use the HAM10000 dataset (Tschandl et al., 2018). For the final feature, we use the first 10000 images of the ISIC archive. This includes multiple data sources but especially over 9000 images from the SONIC dataset presented in Scope et al. (2016). Images from this dataset were also included in the ISIC 2017 challenge dataset (Codella et al., 2018) and the ISIC 2019 challenge dataset. Hence, this bias can be included in all classifiers we investigate in this study.

4.2.2 Application to Adversarial Debiasing

The method described in this work allows us to determine whether a feature is used by a classifier. As a next step, we can use the method during the training of a deep neural network, to force a neural network to avoid using a specific feature. This allows us to train neural networks that ignore a known bias in a dataset and, hence, learn an unbiased classifier from a biased dataset.

The study introduced in this section was originally published in Reimers et al. (2021a). It introduces a novel adversarial debiasing strategy. We layout the basic concept of adversarial debiasing, as well as the main change we propose compared

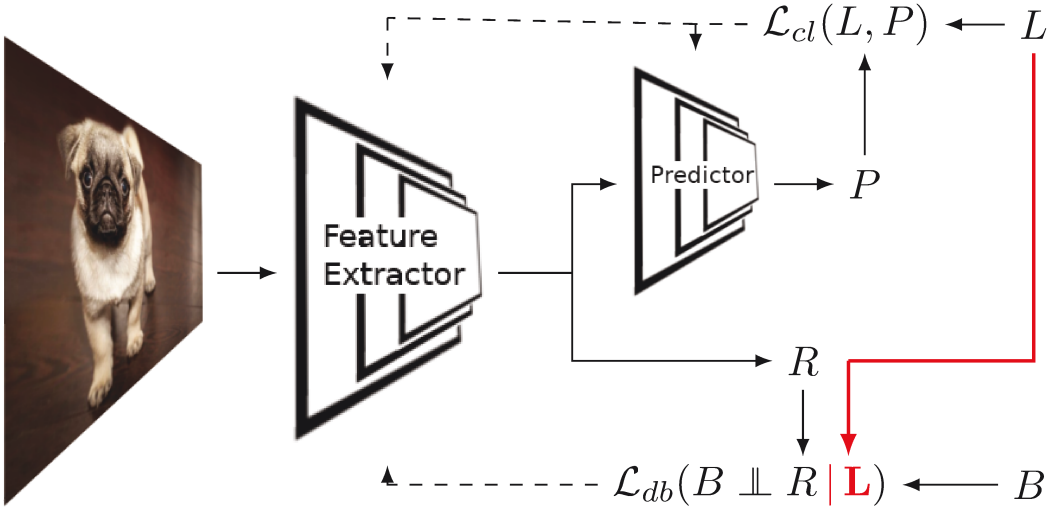


Figure 4.10: *In adversarial debiasing, a debiasing loss $\mathcal{L}_{db}$ is used to enforce independence between the bias variable B and a representation R. In this work, we show that it is beneficial to condition this independence on the label L. This figure was previously published in Reimers et al. (2021a).*

to the state-of-the-art in Figure 4.10. When training a deep neural network for classification, we employ a loss function $\mathcal{L}_{cl}$, for example, the cross-entropy loss to update the weights of the deep neural network towards predicting the correct class. The main idea of adversarial debiasing is to employ a second loss function $\mathcal{L}_{db}$ that penalizes the use of the biased feature. To this end, most authors use a function that quantifies the dependence between the bias variable B and some intermediate representation R from the deep neural network (see Section 4.2.2.1). In contrast, following the method discussed above, we use a loss function that penalizes the conditional dependence between the bias feature B and a representation R derived by the deep neural network given the ground truth label L of the deep neural network.

We start the remainder of this section by introducing the task of adversarial debiasing and explaining the state-of-the-art solutions in Section 4.2.2.1. Of course, like every kind of debiasing, this kind of debiasing is not suitable for every biased situation. We discuss in which situations our method is suitable in Section 4.2.2.2. We introduce three possible implementations of conditional independence as a differentiable loss in Section 4.2.2.3.

Further, in Section 4.3.3, we present experimental evidence that corroborates the claim that our method is better suited for the situation described in Section 4.2.2.2.

4.2.2.1 Adversarial Debiasing

One of the main advantages of deep learning over shallow learning is the automatic feature selection, as described, for example, by Reimers and Requena-Mesa (2020). While the advantages of automatic feature selection are obviously the increase in

accuracy, the drawback is that the user can no longer control which features are used by the classifier to predict the correct class. To this end, the classifier can choose biased features, that in the training data co-occur with a specific label, but are independent of this label in the real world and, consequently, a test set that is representative of the real world.

If we know that our training set contains such a bias, we can employ adversarial debiasing to stop the classifier from picking up the bias and, hence, learn an unbiased classifier from the biased training set. To this end, the classifier is divided into two parts: a feature extractor and a predictor. The feature extractor extracts a representation R of the input. Note that there is no clear choice for where to divide the deep neural network because every neuron can be part of any of the two parts, and it is fair to assume that features of different complexity will be extracted at different layers of the deep neural network (Erhan et al., 2009). Therefore, different choices at which layer we cut the deep neural network and extract the intermediate representation R are reasonable. The most common choice for convolutional neural networks is to use the convolutional layers as the feature extractor and the fully connected layers as the predictor. In addition to classification loss $\mathcal{L}_{cl}$ that pushes the weights of the deep neural network towards predicting the correct classes for all inputs, a second loss is used enforcing that the bias feature B is not used to calculate the representation. The risk in choosing a too early layer is that the bias feature might still be used to calculate the representation, but is dropped later and, hence, does not influence the final prediction. Therefore, in contrast to other works, we use the final prediction of the deep neural network as the representation R. The basic concept of adversarial debiasing can be observed in Figure 4.10.

Based on this concept, authors have to clarify two things to present a adversarial debiasing method. First, we have to choose a criterion to determine whether the classifier uses a feature B. Second, we have to turn this criterion into a differentiable loss.

To the first point, most authors have used either predictability (Zhang et al., 2018a), or statistical dependence (Kim et al., 2019) between the bias feature B and the representation R. To this end, most state-of-the-art adversarial debiasing methods aim to learn a feature representation that is informative for a task but independent of the bias. They train a second neural network to predict the bias variable B from the representation R. The feature extractor of the original network is trained in an adversarial fashion. On the one hand, the classification loss pushes the weights such that the representation R is as informative as possible of the true label L, on the other hand, the debiasing loss pushes the weights of the network such that the representation is as least informative as possible towards the bias variable B. Since the bias variable B and the label L are dependent on the test set, these two goals contradict each other and lead to an adversarial training.

As demonstrated in Section 4, independence is too restrictive as a criterion for determining whether a deep neural network uses a certain feature, as a feature might be dependent on the prediction even if it is not used as demonstrated, for example, in Table 4.4. This fact is further reflected in results from the literature:

Often, the debiased classifiers are less influenced by the bias feature B but also less accurate in predicting the true label L. For example, Alvi et al. (2018) report a significant reduction in bias in an age classifier trained on a dataset biased by gender. However, this reduction in bias is accompanied by a drop in performance on an unbiased test set. The accuracy drops from 0.789 to 0.781. Therefore, we suggest a different criterion to determine whether a neural network uses a feature. We use the conditional independence criterion proposed in Section 4.1.

Our method differs fundamentally from the methods presented in the literature because the criterion to decide whether a bias feature B is used to calculate a representation R is different. Since it diverges in this first aspect, it also diverges in the second aspect of the implementation of these criteria. Other approaches from the literature have proposed different loss functions for the debiasing loss $\mathcal{L}_{db}$. For example, Zhang et al. (2018a) propose to use the mean squared error between the bias variable and the best prediction of the bias variable from the representation. Alvi et al. (2018) suggest minimizing the cross-entropy between the bias prediction and an uniform distribution. Kim et al. (2019) propose using a combination of these two losses and Adeli et al. (2021) suggest minimizing the correlation between the predicted and the observed bias variable.

All of the above loss functions aim to find a representation R that is unconditionally independent of the bias feature, while our approach uses loss functions that aim to find representations that are conditionally independent. To this end, we extend three unconditional dependence methods from the literature. We extend the work of Pérez-Suay et al. (2017) and Li et al. (2019b), which use the Hilbert-Schmidt independence criterion (HSIC) introduced by Gretton et al. (2007) by using the conditional criterion suggested by Fukumizu et al. (2007). We extend the idea of using mutual information presented by Kim et al. (2019) to conditional mutual information and the predictability criterion used, for example, by Adeli et al. (2021) to the maximum correlation coefficient. We offer the details for these three implementations in Section 4.2.2.3.

4.2.2.2 Bias Model

In this section, we discuss for which situations our new adversarial debiasing method is suitable. To this end, we explain the data generation process as a graphical model, that leads to the kind of bias that our method is build to takle. We will further discuss whether this is a common kind of bias in computer vision using examples where the graphical model does or does not fit the data. Further, we present a mathematical proof for a simple situation, showing that the optimal classifier does not satisfy the unconditional dependence which the state-of-the-art uses as a criterion but does satisfy the conditional dependence criterion we suggest.

Many different kinds of bias exist. These different biases influence visual datasets in various ways (Wang et al., 2020). Due to this variety, it is not possible to propose a one fits all solution for debiasing. Hence, in this section, we focus on one specific kind of bias given by a specific, biased data generation process

The graphical model of the data generation for a biased dataset is displayed in

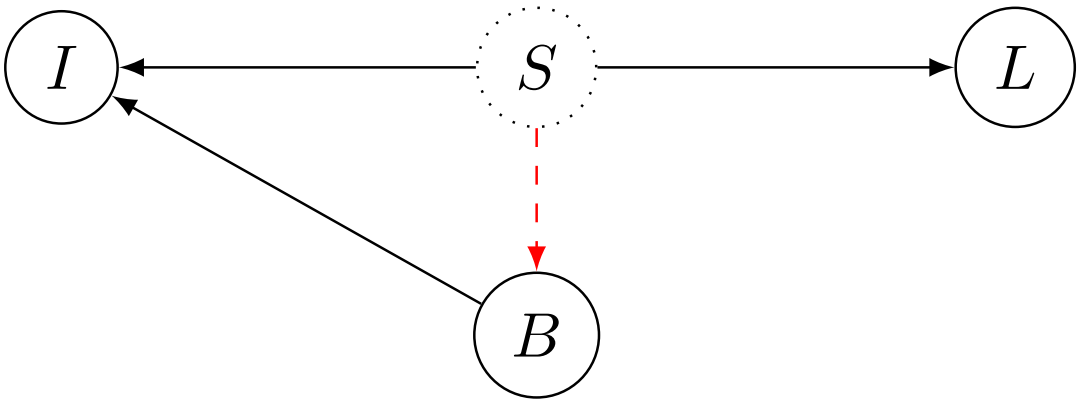


Figure 4.11: *A graphical representation of the specific bias. Circles represent variables, dotted circles represent unobserved variables. The label L is only dependent on a signal S, while the input I is also dependent on some variable B. In the training set, the signal S influences the variable B due to bias. This is indicated by the red dashed arrow. This figure was originally published in Reimers et al. (2021a)*

Figure 4.11. Following this data generation model, we assume that the input images I used in a classification task contain a signal S that can be estimated from the images and should be used to predict the label L. For example, if the classification task is distinguishing between images of cats and dogs, this signal S contains all features that are specific to cats or dogs and, hence, suitable to distinguish them. Consequently, we expect the label of the image to depend only on the signal S. On the other hand, the input image I is influenced by further features beyond the signal S. In our example this includes features like the background or time of day at which the images were taken. These features do not contain information on the class of the object in the image. A bias is introduced into the dataset, if in the dataset, one of these meaningless features is dependent on one of the meaningful features included in the signal S. This can happen due to finite sample effects or because recording specific combinations of meaningful and meaningless features might be dangerous or unethical. The meaningless feature, which in the training set is dependent on the signal S but at test time is independent of the signal S, is called the bias feature B. In our experiments, we, for example, use the fur color of the animal as the bias feature B. Note that, since the fur color is not suitable to distinguish between cats and dogs, it is not part of the signal S but it still influences the image I. Due to our construction of the dataset, the fur color is dependent on whether the image contains a cat or a dog. This dependence can be utilized by a machine-learning approach such as a deep neural network in what is known as a "Clever-Hans" prediction (Lapuschkin et al., 2019), resulting in a biased classifier.

While most of the connections represented by the arrows in the graphical model in Figure 4.11 are obvious, we want to offer additional information on the arrow from S to B. First, we want to explain the consequence of the arrow being in this direction and, afterward, we explain, why we think that this direction of the arrow is correct for most tasks in computer vision.

The arrow from S to B indicates the main difference between the train and the test set. By orienting the arrow in this direction, we express the fact, that we expect the distribution of the signal S to be the same during training and test. In contrast, since one of the influences is eliminated at test time, we accept that properties of the distribution of B will differ between the training set and the test set. We consider this a realistic situation, as we would expect a researcher to ensure that the important signals in the training set are a faithful representation of these signals in the real world, while we expect the researcher to pay less attention to features that are not meaningful towards classifying the object in the image.

Towards the second point, we want to emphasize that the data used in a classification task is not selected randomly, but a researcher will carefully select it. The main criterion used to decide, whether to include an image, will always be the signal S that contains information on the true label L. A researcher will ensure that the distribution of this signal matches the distribution of this signal in the real world and will, for this purpose, accept a misrepresentation of the distribution of B. On the other hand, a researcher will not accept a mismatch between the true distribution of S and its representation in the training set to model the distribution of B correctly. Therefore, S influences B through the dataset acquisition while B does not influence S through this process.

While this data generation model covers many relevant real world applications of computer vision, no single model can cover all situations. To this end, we give two examples where it does hold, and one example where it does not hold.

The first example is a driver assistance system that warns the driver of an aquaplaning risk using a camera image as presented, for example, in Schneider et al. (2018). To train a neural network for this task, we require a training set of images containing situations with different levels of aquaplaning risk. Images of situations of low aquaplaning risk can easily be collected on various streets in the wild. In contrast, driving in aquaplaning conditions is dangerous. Hence, images of situations that depict a high aquaplaning risk have to be taken in a specific facility where it is safe to drive into aquaplaning conditions. In this example, standing water on the street is the signal S. The bias feature B in this example is the location, which determines facts like the tarmac and the road markings in the image. Due to the safety risk, these two signals are dependent in the training set but not during the application of the system in the real world.

The second example is a system that automatically classifies skin lesions from images as described, for example, by Tschandl et al. (2018). To train a deep neural network we rely on images taken by practitioners. An important feature to determine whether the skin lesion is dangerous is the growth of the skin lesion over time. Hence, if a practitioner suspects a skin lesion to be cancerous, they draw a scale next to the skin lesion to compare the size during subsequent checkups. Hence, the probability to be cancerous influences the probability of a scale drawn next to the skin lesion. In this example, the medical features of the skin lesion form the signal S and the drawn scales are the bias feature B. These two signals are dependent in the training set, but we expect them to be independent during the application of an

automatic skin lesion classifier in the wild. More examples of biases in this setting can be found in Section 4.2.1.

These examples demonstrate that the data generation model fits different bias situations in machine learning. However, since the possible biases are plentiful, no single method can fit all bias situations. We find, that our method is not suited for some situations in algorithmic fairness, since the connection between the meaningful and the meaningless features prevails even during the application of the method. For further explanations, we provide the following example.

This example concerns a system that predicts absenteeism from work as proposed, for example, by Ali Shah et al. (2020). If a predictor is tasked to predict absenteeism from work, the result might be biased against women, as they can be absent from work due to pregnancy. In this example, the signal S consists of personal reasons to be absent from work and the bias feature B is the sex of the person. Due to pregnancy, these two are dependent in the training set, but, in contrast to the situation described above, this dependence is also present during the application of the method.

The fact that we rigorously define the bias model allows us to investigate the advantages of our proposed solution analytically. Under the assumption of the bias model, the optimal classifier fulfills the conditional dependence

$$R \perp\!\!\!\perp B \,|\, L \tag{4.65}$$

but does not fulfill the unconditional dependence

$$R \perp\!\!\!\perp B. \tag{4.66}$$

In this work, we only include a proof for a simple linear, univariate case. More specifically, all arrows in the graphical model described above describe linear functions and the label is univariate. We believe that this proof can be extended to the non-linear case by substituting the scalar product with the scalar product in a suitable kernel space. Furthermore, it can be extended to the multidimensional case by multiplying with the inverse of the matrix $\langle L, L \rangle$ instead of dividing by it.

Theorem 2. *If the bias can be modeled as displayed in Figure 4.11, the optimal classifier fulfills the conditional independence in (4.65) but not the independence in (4.66).*

Proof of Theorem 2. This proof is taken verbatim from Reimers et al. (2021a) and the supplementary material of that paper.

Throughout this proof, we denote all variables with capital Latin letters (C, L, R, S). Capital Greek letters denote processes (Φ, Ψ, Ξ). For these processes, we denote the linear coefficients with lower-case Greek letters (α, ζ). The only exception to this is the optimal classifier that is denoted by F^*.

We start the proof by defining all functions involved in the model. Afterward, since dependence results in correlation in the linear case, a simple calculation proves the claim. Let S denote the signal according to the bias model, as explained

above. Since we are in the linear case, the bias variable B can be split into a part that is fully determined by S and a part that is independent of S.

Let B^* be the part of the bias variable that is independent of S. The bias variable B is given by

$$B = \alpha_1 S + \alpha_2 B^* =: \Phi(S, B^*). \tag{4.67}$$

Further, the label L can be calculated from the signal S

$$L = \zeta_1 S =: \Xi(S) \tag{4.68}$$

and the image I is given by

$$I =: \Psi(S, B) = \Psi(S, \Phi(S, B^*)). \tag{4.69}$$

The optimal solution F^* of the machine learning problem will recover the signal and calculate the label. By the assumptions of the bias model, the signal can be recovered from the input. Thus, there exists a function $\Psi^\dagger$ such that

$$\Psi^\dagger(\Psi(S, B)) = S \tag{4.70}$$

holds. Therefore, F^* is given by

$$F^* := \Xi\Psi^\dagger. \tag{4.71}$$

Now, we have defined all functions appearing in the model. The rest of the proof are two straightforward calculations. In the linear case, the independence of variables is equivalent to variables being uncorrelated. We denote the covariance of two variables A, B with $\langle A, B\rangle$. To prove that (4.66) does not hold, we calculate

$$\begin{aligned}\langle F^*(I), B\rangle &= \left\langle \Xi\Psi^\dagger\Psi(S, \Phi(S, B^*)), \Phi(S, B^*)\right\rangle \\ &= \langle \zeta_1 S, \alpha_1 S + \alpha_2 B^*\rangle = \zeta_1\alpha_1 \langle S, S\rangle.\end{aligned} \tag{4.72}$$

This is equal to zero if and only if either all inputs contain an identical signal ($\langle S, S\rangle = 0$), the dataset is unbiased ($\alpha_1 = 0$), or the label does not depend on the signal ($\zeta_1 = 0$).

For the conditional dependence, we compute the partial correlation. By definition of the partial correlation, we find

$$\langle F^*(I), B\rangle | L = \langle F^*(I) - \widehat{F^*_I}(L), B - \widehat{B}(L)\rangle. \tag{4.73}$$

Here, $\widehat{F^*_I}(L)$ is the best linear regression of F^* given L, and $\widehat{B}(L)$ is the best linear regression of B given L.

Now we use the formula to calculate the linear regression coefficient and write the best linear regression as the multiplication of the coefficient and the value L to get

$$\widehat{F^*_I}(L) = \frac{\langle F^*(I), L\rangle}{\langle L, L\rangle} L \tag{4.74}$$

and

$$\hat{B}(L) = \frac{\langle B, L\rangle}{\langle L, L\rangle} L. \tag{4.75}$$

We plug these results into (4.73) to get

$$\langle F^*(I), B\rangle | L = \langle F^*(I) - \frac{\langle F^*(I), L\rangle}{\langle L, L\rangle} L, B - \frac{\langle B, L\rangle}{\langle L, L\rangle} L\rangle. \tag{4.76}$$

We expand the scalar product to find

$$\begin{aligned} &\langle F^*(I), B\rangle | L = \\ &\langle F^*(I), B\rangle - \langle F^*(I), \frac{\langle B, L\rangle}{\langle L, L\rangle} L\rangle - \langle \frac{\langle F^*(I), L\rangle}{\langle L, L\rangle} L, B\rangle + \langle \frac{\langle F^*(I), L\rangle}{\langle L, L\rangle} L, \frac{\langle B, L\rangle}{\langle L, L\rangle} L\rangle \\ &\qquad = S_1 - S_2 - S_3 + S_4. \end{aligned} \tag{4.77}$$

Simplifying S_2 and S_3 individually by pulling the fraction out of the scalar product, we get

$$S_2 = \frac{\langle F^*(I), L\rangle \langle B, L\rangle}{\langle L, L\rangle} \tag{4.78}$$

and

$$S_3 = \frac{\langle B, L\rangle \langle F^*(I), L\rangle}{\langle L, L\rangle}. \tag{4.79}$$

For S_4 we pull both fractions out of the scalar product and reduce the resulting fraction by $\langle L, L\rangle$ to get

$$S_4 = \frac{\langle F^*(I), L\rangle \langle B, L\rangle}{\langle L, L\rangle^2} \langle L, L\rangle = \frac{\langle B, L\rangle \langle F^*(I), L\rangle}{\langle L, L\rangle}. \tag{4.80}$$

As we find

$$S_2 = S_3 = S_4 \tag{4.81}$$

we conclude

$$\langle F^*(I), B\rangle | L = S_1 - S_2 = \langle F^*(I), B\rangle - \frac{\langle F^*(I), L\rangle \langle B, L\rangle}{\langle L, L\rangle}. \tag{4.82}$$

Now we substitute L by $\zeta_1 S$, $F^*(I)$ by $\zeta_1 S$, and B by $\alpha_1 S + \alpha_2 B^*$ to get

$$\langle F^*(I), B\rangle = \langle \zeta_1 S, \alpha_1 S + \alpha_2 B^*\rangle = \zeta_1 \alpha_1 \langle S, S\rangle + \zeta_1 \alpha_2 \langle S, B^*\rangle. \tag{4.83}$$

Since B^* and S are independent by definition of B^*, we can evaluate $\langle S, B^*\rangle = 0$ and find

$$\langle F^*(I), B\rangle = \zeta_1 \alpha_1 \langle S, S\rangle. \tag{4.84}$$

Similarly, we find

$$\begin{aligned}\frac{\langle F^*(I), L\rangle\langle B, L\rangle}{\langle L, L\rangle} &= \frac{\langle \zeta_1 S, \zeta_1 S\rangle\langle \alpha_1 S + \alpha_2 B^*, \zeta_1 S\rangle}{\langle \text{`}\zeta_1 S, \zeta_1 S\rangle} \\ &= \frac{\zeta_1^2\langle S, S\rangle(\alpha_1\zeta_1\langle S, S\rangle + \alpha_2\zeta_1\langle B^*, S\rangle)}{\zeta_1^2\langle S, S\rangle}.\end{aligned} \tag{4.85}$$

We use the fact that $\langle S, B^*\rangle = 0$ by definition of B^* to find

$$\frac{\langle F^*(I), L\rangle\langle B, L\rangle}{\langle L, L\rangle} = \frac{\zeta_1^2\langle S, S\rangle\alpha_1\zeta_1\langle S, S\rangle}{\zeta_1^2\langle S, S\rangle}. \tag{4.86}$$

Reducing this fraction by $\zeta_1^2\langle S, S\rangle$ leads to

$$\frac{\langle F^*(I), L\rangle\langle B, L\rangle}{\langle L, L\rangle} = \alpha_1\zeta_1\langle S, S\rangle. \tag{4.87}$$

Finally, we include the results of (4.84) and (4.87) into (4.82) to arrive at

$$\langle F^*(I), B\rangle | L = \alpha_1\zeta_1\langle S, S\rangle - \alpha_1\zeta_1\langle S, S\rangle = 0. \tag{4.88}$$

This concludes the proof. □

A consequence of this theorem is that the optimal classifier does not fulfill the independence (4.66) and hence, will not minimize any loss function based on this criterion. Furthermore, Equation (4.72) tells us that the unconditional correlation is given by $\zeta_1\alpha_1\langle S, S\rangle$. To reduce this value, a neural network will try to ignore the connection indicated by ζ_1, which is the connection between the signal S and image I. Hence, an unconditional dependence loss will drive the classifier towards weakening the crucial connection between the relevant signal S and its prediction P. Hence, the unconditional criterion will reduce the performance of a classifier on an unbiased test set. This behavior was reported in the literature, for example, Alvi et al. (2018), report a significant reduction in bias in an age classifier trained on a dataset biased by gender. However, this reduction in bias is accompanied by a drop in performance. On an unbiased test set, the accuracy droped from 0.789 to 0.781. And it is, further, what we found in our experiments, which we report in Section 4.3.3.

In comparison, the optimal classifier fulfills the conditional dependence criterion (4.65) and, therefore, will minimize losses based on this criterion. Thus, using a loss function based on this criterion will not reduce the performance of the classifier.

4.2.2.3 Implementation

The previous section explains why a loss based on our new conditional independence is better suited for the bias situation depicted in Figure 4.11 than a loss based on the traditional unconditional dependence criterion. However, conditional independence tests are binary in their result, and hence not differentiable, a loss

function, in contrast, differentiable. To this end, we use the test statistic of different conditional independence tests as a loss function. For conditionally independent variables, the test statistics of the conditional independence test are close to zero. More specifically, they would be zero for an infinite sample, but will be slightly larger due to finite sample effects. Hence, we can use the test statistic as a loss function.

Further, for a conditional dependence test, we need to approximate the joint distribution of the prediction P, the bias variable B and the true label L. During the training, however, we see only a mini-batch of examples. To counteract this fact, we use momentum on the optimizer to average the gradients across multiple mini-batches.

In this section we present three conditional independence tests and describe how to calculate the test statistics. This three conditional independence tests are the conditional mutual information (MI), the Hilbert-Schmidt conditional independence criterion (HSCONIC) and the maximum partial correlation criterion (MCC). We introduce these tests in Section 2.3.

Conditional Mutual Information Criterion: The first conditional dependence test we use is the conditional mutual information. The conditional mutual information is the conditional version of the mutual information. The mutual information between the predictions R and B is given by

$$\mathrm{MI}(R;B) = \sum_{r\in\mathcal{R},b\in\mathcal{B}} p_{R,B}(r,b)\log\frac{p_{R,B}(r,b)}{p_R(r)p_B(b)}. \tag{4.89}$$

The mutual information is zero if and only if the two variables are independent. Hence, the mutual information was used as a loss for adversarial debiasing by Kim et al. (2019). Since we are looking for conditional independence, we suggest using conditional mutual information instead. The conditional mutual information between the prediction R and bias variable B given the label L is defined as

$$\mathrm{MI}(R;B|L) = \sum_{l\in\mathcal{L},b\in\mathcal{B}r\in\mathcal{R}} p_{R,B,L}\,(r,b,l)\log\frac{p_L(l)p_{R,B,L}\,(r,b,l)}{p_{R,L}(r,l)p_{B,L)}}. \tag{4.90}$$

It is zero if and only if the prediction R and B are independent given the label L. However, to evaluate the conditional mutual information we have to calculate the densities $p_{R,B}, p_R$ and p_B which is difficult. We use kernel density estimation on the mini-batches to determine the densities. We employ a Gaussian kernel with a variance of one-fourth of the average pairwise distance within a mini-batch. This setting proved best in preliminary experiments on reconstructing densities.

Hilbert-Schmidt Conditional Independence Criterion: The Hilbert-Schmidt independence criterion was introduced by Gretton et al. (2007). The main idea is to create a kernel matrix that contains the pairwise similarities between samples. Then, for the observed pairs of prediction R and the bias variable B, we quantify whether the prediction R is close to predictions of the same pairs for which the bias

variable is close to bias variable B. More specifically this can be calculated by

$$\text{HSIC}(R, B) = \frac{1}{(m-1)^2} \text{tr}\, K_R H K_B H. \tag{4.91}$$

Here, K_R and K_B denote the kernel matrices for R and B, respectively. These kernel matrices contain the pairwise similarities between examples. For the Kronecker-Delta δ_{ij} and m the number of examples, H is given by $H_{ij} = \delta_{ij} - m^{-2}$. We explain the idea behind the Hilbert-Schmidt independence criterion (HSIC) in Section 2.3.4. The test statistic of the HSIC was suggested as an additional condition for optimization in classical machine learning methods by, for example, Pérez-Suay et al. (2017) and Li et al. (2019b). In contrast to our method, however, these classical machine learning approaches do not rely on gradient descent and, hence, do not have to solve the problem of differentiability and the problem to estimate the test statistic on mini-batches.

For our method, we need a loss that enforces conditional independence. For this reason, we use the test statistic of a Hilbert-Schmidt conditional independence criterion,

$$\text{HSCONIC}(R, B \mid L) \approx \text{tr}\, G_R S_L G_B S_L. \tag{4.92}$$

It is equal to zero if and only if the variables R and B are conditionally independent given L. Here, S_L is given by $(\mathbb{I} + 1/m G_L)^{-1}$, where $\mathbb{I}$ is the identity matrix and $G_X = H K_X H$ with K_X the kernel matrix for $X \in \{B, R, L\}$ and $H_{ij} = \delta_{ij} - m^{-2}$ for δ_{ij} the Kronecker-Delta and m the number of examples. We use the same kernel as above and estimate the loss on every mini-batch independently.

Maximum Partial Correlation Coefficient Criterion: A common idea in adversarial debiasing is to penalize the predictability of the bias variable B from the intermediate representation R measured, for example, by the mean squared error

$$\min_f \int (b - f(r))^2 \, dp_{B,R}(b, r). \tag{4.93}$$

This course of action was, for example, suggested by Zhang et al. (2018a). We describe this idea and its relation to the maximum correlation coeficient in Section 2.3.5. As described in that section, a form of this predictability criterion is given by

$$\max_f \text{Corr}\,(B, f(R))^2\,. \tag{4.94}$$

As an extension of this criterion, we use the maximum correlation criterion that we introduce in Section 2.3.2. This criterion uses the following maximum as a test statistic,

$$\text{MCC}(R, B) = \max_{f,g} \text{Corr}(f(R), g(B)). \tag{4.95}$$

To turn this into a conditional independence test, we use the partial correlation

$$\text{MPCC}(R, B \mid L) = \min_{h} \max_{f,g} \text{PC}(f(R), g(B) \mid h(L)). \tag{4.96}$$

Note that in the classification setting we try to predict a discrete label in a metric space that uses the trivial metric. This metric assigns zero as the distance between a label and itself and a constant value to the distance between each pair of different labels. A typical metric space that fulfills this requirement is one-hot encoded class labels. In this setting every function is a linear function and, hence, we can omit the function h to get

$$\text{MPCC}(R, B \mid L) = \max_{f,g} \text{PC}(f(R), g(B) \mid L). \tag{4.97}$$

To parameterize both functions f and g, we use neural networks.

To disentangle the effects of both of these changes, the change from the predictability criterion used by Zhang et al. (2018a) and Adeli et al. (2021) to the maximum correlation criterion and the change from the maximum correlation criterion to the maximum partial correlation criterion, we present an ablation study in Section 4.3.3.2. In that section, we compare four loss function. First, the original loss function used by Adeli et al. (2021) which we displayed in (4.94). Second, we use the maximum correlation coefficient, which can be found in (4.95). The maximum correlation coefficient includes the change to the maximum correlation but not the change to partial correlation. Hence, third, we compare the contrary situation which includes the change to the partial correlation but not the change to the maximum correlation coefficient. In this situation, the loss is given by

$$\max_{f} \text{PC}(f(R), B \mid L). \tag{4.98}$$

Finally, we compare these loss functions to the one that includes both changes, the change to partial correlation and the change to maximum correlation. This loss function is displayed in equation (4.97).

All three of these implementations can be used for vector-valued variables. Therefore, they can also be used to de-bias from multiple bias variables at the same time.

4.3 Experiments

In this section, we demonstrate the empirical evidence to support the claims made in the previous sections. We start with experiments to show that our method is suitable to determine whether a classifier uses a feature in Section 4.3.1. This includes a study on automatic skin lesion classifiers in Section 4.3.2. Finally, we present experiments that demonstrate that our method can be used for adversarial debiasing in Section 4.3.3.

4.3.1 Experiments to Demonstrate that the Method is Applicable

In this section, we provide a proof of concept for our method and evidence that it can be applied to real-world classifiers and find correct answers. Therefore we present a synthetic data example where we know the feature usage of different classifiers beforehand in Section 4.3.1.1. Then, we show how to use it to compare classifiers concerning specific properties without a specialized dataset. In particular, in Section 4.3.1.2, we present a comparison between two classifiers on the MS COCO dataset (Lin et al., 2014). We compare two classifiers concerning their dependence on the position of objects in the image. Since these classifiers detect 80 different classes, creating a dataset, where each object appears even approximately in every position equally often is infeasible. Our method can be used to compare these two classifiers directly. Finally, our method can be used to determine more specific information about features that have been discovered using explorative methods or to evaluate hypotheses determined by other methods. In Section 4.1.7.1, we introduced an example, where an area is highlighted but we do not know which property of this area is important. Following this discussion, we use a classifier trained to distinguish birds. In Section 4.3.1.3, we demonstrate how our method can be used to test which property of the area is considered by the deep neural network.

4.3.1.1 A Synthetic Example

The first experiment we present for the method demonstrated in this work was originally published in Reimers et al. (2019). The goal of this experiment is to demonstrate that the method can identify the correct features used by different prediction methods. We start with a short description of the synthetic dataset. Afterward, we present the features we consider in this study and describe the different predictors we use in this experiment.

The inputs to the prediction task consist of mono-color 8×8 images A. The values $a_{i,j}$ of their pixels are independently identically distributed following a normal distribution

$$a_{i,j} \sim \mathcal{N}(\alpha_A, \beta_A^2). \tag{4.99}$$

The task is to estimate the parameter β_A from the image. We created 20 000 examples where both parameters α and β are uniformly distributed on the interval $[0, 1]$.

The two features, which we use as the feature of interest are the sample mean and the sample standard deviation. The sample mean α^* is given by

$$\alpha^* = \frac{1}{64} \sum_{i,j=1}^{8} a_{i,j} \tag{4.100}$$

and the sample standard diviation β^* is given by

$$\beta^* = \sqrt{\frac{1}{64}\sum_{i,j=1}^{8}(a_{i,j} - \alpha^*)}. \tag{4.101}$$

Of course, the first feature is close to useless in predicting β, while the second feature is near perfect.

We compare four predictors for this task. The first predictor, which we call F_1, is the sample standard deviation

$$F_1(A) = \sqrt{\frac{1}{64}\sum_{i,j=1}^{8}(a_{i,j} - \alpha^*)} = \beta^*. \tag{4.102}$$

This predictor obviously uses the second feature β^* but is mathematically independent of the feature α^*.

The second predictor we use in this experiment is given by a fully convolutional neural network. This predictor is called F_2. This network has three convolutional layers with 4, 16 and 64 filters of size 2×2 and a stride of 2×2. After the three layers, we are left with a representation of size $1 \times 1 \times 64$. To reach our final prediction, we use a final convolutional layer with 1 filter of size 1×1. We use regularized linear units as the activation in all but the last layer. We trained the network using the Tensorflow framework (Abadi et al., 2015). As the optimizer, we use stochastic gradient descent for 200 000 steps with a learning rate of $3e^{-5}$.

The third predictor is a simple linear classifier on the pixel values of A. It is called F_3. Note that, since $F_3(A)$ is a linear combination of the $a_{i,j}$ it is independent of the feature β^*. We expect that this predictor will over adapt to some finite data effect in the training set and will hence make use of the linear feature α^* but not the quadratic feature β^*.

Finally, the fourth predictor F_4 is an oracle classifier. This classifier is independent of the input and, hence, it uses no feature, in particular not the feature α^* and the feature β^*.

As the independence test in this experiment, we use the fast conditional independence test (FCIT) presented by Chalupka et al. (2018) and discussed in Section 2.3.2. For this conditional independence test, we need to determine the joined distribution of the feature, the ground truth parameter and the prediction of the predictor F_i:

$$(\alpha, \alpha^*, F_i) \qquad \text{and} \qquad (\beta, \beta^*, F_i). \tag{4.103}$$

To this end, we use a test set of 10 000 images not used during training.

We first evaluate the quality of the four predictors. For each of the four predictors, we calculated the mean squared error between the ground truth and the prediction measured on the 10 000 test images. We report these values in the second column of Table 4.10. The oracla predictor F_4 reaches a perfect mean squared error of zero. The linear predictor F_3 achieves a mean square error of 0.084 which is worse than

Table 4.10: *Results of the experiment for all four estimators. We report the mean squared error (MSE) between the ground truth and the prediction on the test set. Further, we report whether the features α^* and β^* are used. We report the p-values from the FCIT. We assume significance at a level of 0.01. Significant results are marked in bold. The content of this table was previously presented in Reimers et al. (2019)*

Predictor	MSE	α^*		β^*	
		p-value	sig.	p-value	sig.
F_1	0.0026	0.5137	No	**0.0014**	**Yes**
F_2	0.0153	0.4576	No	**0.0091**	**Yes**
F_3	0.0840	**0.0052**	**Yes**	0.3710	No
F_4	0	0.05912	No	0.0728	No

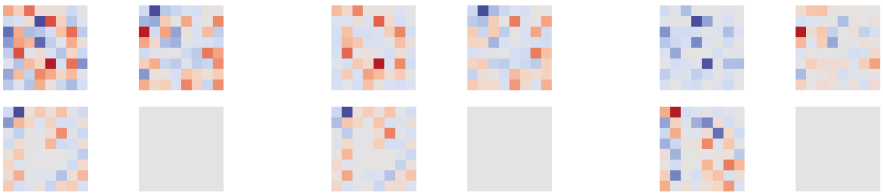


Figure 4.12: *In every sub-figure, the results of one saliency method are displayed. The left is based on the gradient (compare Section 2.4.1.1), the right one is based on occlusion (compare Section 2.4.1.2) and the middle one on the combination (compare Section 2.4.1.3). The top line of every sub-figure shows the results for estimators F_1 and F_2, in the second line the results for estimators F_3 and F_4. Since the oracle classifier F_4 does not depend on the inputs, the results for F_4 is always zero for all inputs.*

a constant predictor that would have reached a mean squared error of 0.083. This demonstrates, that the linear predictor provides almost no information on β^*. The predictor F_1 that is built to use feature β^* reaches a mean squared error of 0.003 which is 32 times smaller than the linear predictor. Finally, the neural network reaches an error of 0.015. This shows that it can solve the task much better than the linear classifier.

For the use of features, we find that our method correctly determined that the oracle predictor F_4 does not use any of the features. It, further, determined that the hand-crafted predictor F_1 uses only the feature β^* but not the feature α^*. The result for the two trained predictors is also as expected. The linear predictor F_3 uses only the linear feature α^* but not the quadratic feature β^*. This is in accordance with the theory and the large mean squared error. In contrast, the neural network uses only the quadratic feature β^* and not the useless, linear feature α^*.

To demonstrate that our method is a valuable addition to other methods, for example, saliency maps, in Figure 4.12, we display the output of three saliency map methods: One based on the gradients of the classifier (compare Section 2.4.1.1), one

based on occlusion (compare Section 2.4.1.2) and one based on the combination (compare Section 2.4.1.3). We argue that none of these outputs can be used to determine whether feature α^* or β^* is used by any classifier other than F_4.

4.3.1.2 Evaluating and Comparing Classifiers on the MS COCO Dataset

In the second experiment, we present evaluations and comparisons of complex real-world classifiers on the MS COCO dataset (Lin et al., 2014). The MS COCO Dataset is a multi-label multi-class dataset of real-world images. We present this experiment to demonstrate three things: First, we want to demonstrate that the method presented in this work can be applied to complex real-world datasets. Second, we demonstrate that the method can be applied to complex, pre-trained classifiers in a black-box setting. Third, we demonstrate how to use it to compare classifiers beyond test set accuracy. This experiment was originally presented in Reimers et al. (2020).

The first classifier we use for these experiments is the Multi-Label Graph Convolutional Networks (ML-GCN) presented by Chen et al. (2019). The second classifier is the Spatial Regularization with Image-level Supervision classifier (SRN) presented by Zhu et al. (2017). For both of these classifiers, we relied on the code provided by the authors. For the ML-GCN we, in addition to the code, used the weights provided by the authors.

For this dataset and classifiers we investigate multiple features. The first feature is the area of the object. We expect a classifier to be more sure the bigger the object in the image is. To quantify the area we calculate the fraction of the image that is used up by the object. If multiple instances of the object exist in the image we add all there areas. To determine the area we use the ground truth image segmentation provided with the dataset.

One of the differences between MS COCO and other datasets, for example, ImageNet (Russakovsky et al., 2013) is that MS COCO has less bias towards the center. While in ImageNet, the important object is most often in the center, in MS COCO it can appear anywhere in the image. Hence, all additional features we investigate concern the position of the object in the image.

Many classifiers, including the ML-GCN, are trained using data augmentations. These data augmentations include horizontal flips of images. Hence, we are especially interested in the difference between the horizontal and the vertical position of the object in the image. The features we investigate are:

- whether the object appears in the top or bottom half of the image,
- whether the object appears in the left or right half of the image,
- the x and y coordinates of the object in the image,
- the angle between the first axis of the image and the line between center of the image and center of the object.

For the position, we consider the center of mass of all instances of the object in the image.

Table 4.11: *The results of the experiments on the MS COCO dataset for the different classifiers. For each feature and each classifier, we report, for how many out of the 80 classes, the feature was relevant. This table was originally published in Reimers et al. (2020)*

Feature	ML-GCN (Chen et al., 2019)	SRN (Zhu et al., 2017)
Area	73/80	69/80
Half(Horizontal)	1/80	1/80
Position(Horizontal)	10/80	16/80
Half(Vertical)	25/80	32/80
Position(Vertical)	51/80	54/80
Angle	24/80	31/80

To test for independence, we use the Randomized conditional Correlation Test (RCOT) presented in Strobl et al. (2019) and we explain it in Section 2.3.4. Since we need the joint distribution of feature, prediction and ground truth, we approximate it using the validation set of the 2014 challenge data set. We perform many independent tests. Therefore, we use a significance level of 0.001.

The results of our dependence tests are reported in Table 4.11. The MS COCO dataset contains labels for 80 classes. In the table, we report for how many of these classes we found a significant dependence.

We found that, for most classes, the area of the object is relevant for the detection score. The only classes for which the area is irrelevant are “sheep,” “elephant,” “bear,” “giraffe,” “kite,” “toaster,” and “hairdryer”. In contrast to an average object, which appears in 1267 images of the dataset, instances of “hairdryer” and “toaster” appear in only less than 75 images. This might lead to the network mostly relying on context to recognize objects of these two classes and, hence, performs better if the context is clearer and not if the object is bigger. Of the five remaining classes, interestingly, four are animals.

For the positional features, we find that despite the horizontal flipping, for one class, namely “mouse,” it is relevant to both classifiers in which half of the image the object appears. Other than this, the results for the positional features are as expected. The vertical position/half is more often used than the horizontal position and the more informative features (Position) are used more often then the less informative feature (Half).

In comparison, we find that the ML-GCN relies on the area of the object for more objects and on the position for less classes then the SRN. Since we want the ideal classifier to rely on the objet and be independent of the position of the object, we can use this analysis to compare the classifiers. In this case we prefere the ML-GCN over the SRN.

Figure 4.13: *From left to right: An example of a white-crowned sparrow, an example from a white-throated sparrow, the segmentation map for the white throat markings, the segmentation map for the yellow lores. This figure was originally presented in Reimers et al. (2020).*

Table 4.12: *Results for the independence test that determines whether the yellow lores or the white throat markings are relevant to the LSFGC to distinguish white-crowned sparrows from white-throated sparrows. These results were originally presented in Reimers et al. (2020). Significant results are marked in bold*

Feature	Area		Color	
	p-value	sig.	p-value	sig.
Yellow lores	0.406	No	0.446	No
white throat markings	0.404	No	0.330	No

4.3.1.3 Investigating a Classifier on CUB200

The next experiment we conduct is to demonstrate that our method can be used to make fine-grained investigations into which feature is used. This experiment was originally reported in Reimers et al. (2020). The classifier we investigate here is the Inception-V3 large scale fine-grained classifier (LSFGC) from Cui et al. (2018) that is pre-trained on the iNaturalist 2017 dataset (Van Horn et al., 2018) and fine-tuned on CUB200 (Welinder et al., 2010). We employ the LSFGC as a two-class classifier, that only differentiates between two bird species, the white-crowned sparrow (Zonotrichia leucophrys) and the white-throated sparrow (Zonotrichia albicollis).

In Section 4.1.7.1, we showed an example for a classifier that considers the yellow lores of these bird to be relevant. However, this is unspecific as a feature. Hence, we compare two features for the yellow lores and the white throat markings, the area and the color of these regions.

For the area, we measure which fraction of the image is covert by the respective part of the bird, for the color feature, we average the hue of all pixels in the area covert by the respective part of the bird, by converting the image into an HSV color space. Segmentation maps for both of these features can be found in Figure 4.13.

To determine independence, we again use the RCOT, compare Section 2.3.4. We assume independence at a p-value below 0.05. The results are displayed in Table 4.12.

For no feature, we find a strong enough dependence to reject the null-hypothesis of independence. This demonstrates that deep neural networks in general do not use the same features that humans use. And instead rely on features that might be

difficult to interpret. This is in accordance with adversarial examples as described by Szegedy et al. (2013) and Goodfellow et al. (2014b) and discussed in Section 3. The network instead focusses on many small differences that provide complex patterns that identify these birds.

These experiments only demonstrate the general use of the method and show that they can be applied to and return meaningful results for real-world deep neural network classifiers. We demonstrated that it can be used for complex classifiers without any retraining or extraction of intermediate results and on features that can not easily be manipulated without changing integral parts of the image or for which a derivative can be calculated.

While this is only a proof of concept, we demonstrate a real-world use case, in which we apply the method to the relevant problem of automatic skin lesion classification in Section 4.3.2.

4.3.2 Experiments on Skin Lesion Classification

Tables 4.13, 4.14, 4.16 and 4.15 contain the results of the experiments for the four groups of features explained in Section 4.2.1.4. We divide the rows of these tables into three groups. The first group contains the models following Perez et al. (2018) that we trained for melanoma recognition. The second group contains the models following Perez et al. (2018) that we trained to recognize seborrheic keratosis. The third group is the models following Gessert et al. (2020). We abbreviate the different models in the following way: First, we denote whether the model follows Perez et al. (2018) by "Per" or Gessert et al. (2020) by "Ges". For the models following Perez et al. (2018), we then denote the backbone network (ResNet-152 (R), Inception-v4 (I), DenseNet-161 (D)) the aggregation method (mean (n), maximum (x)), the number of augmented samples and the class it is trained to recognize. For example, Per:Dx26::MEL is a DenseNet-161 trained following Perez et al. (2018) for melanoma recognition using maximum aggregation of 26 augmented samples. For the ensemble classifier following Gessert et al. (2020) we simply denote the class it is trained to recognize.

We report the results of three different dependence tests, namely PC, FCIT, and HSIC (compare Section 2.3). If we say a feature is used by a classifier, we mean that a majority of dependence tests indicated that the feature was conditionally dependent on the prediction.

We first report the results for the four features that have little to no information on the skin lesion in Table 4.13. No test indicates that the orientation of the skin lesion is used. This is promising since it contains no information on the skin lesion at all. Overall, the majority of tests indicated the use of a feature for only four combinations of classifier and feature out of the 128 combinations tested in this group of features. The results of these experiments match the expectation for this experiment. The features are not, or very rarely used by any classifier which increases our trust in the correctness of the positive results found in the following experiments.

Table 4.13: *Results of the validation of the conditional dependence method. For every classifier and feature/test, we indicate that the feature is used with a ✓ or not used with an ✗. We assume a feature is used if the test reports a significant dependence at $p = 0.01$. The models of Perez et al. (2018) are denoted by "Per" followed by their backbone (ResNet-152 (R), Inception-v4 (I), DenseNet-161 (D)) the aggregation method (mean (n), maximum (x)) and the number of augmented samples. Models from the ensemble from Gessert et al. (2020) are denoted by "Ges" followed by the predicted class as melanoma (MEL), melanocytic nevus (NV), basal cell carcinoma (BCC), actinic keratosis (AK), benign keratosis (BKL), dermatofibroma (DF), vascular lesion (VASC) and squamous cell carcinoma (SCC). The star (*) denotes cases where the labels already explain all of the observed variance. These results were originally reported in Reimers et al. (2021b)*

Classification	Orientation			Rand. Symmetry			Image ID			MNIST Class		
Model	PC	FCIT	HSIC	PC	FCIT	HSIC	PC	FCIT	HSIC	PC	FCIT	HSIC
Per:Dx26::MEL	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Per:Dn26::MEL	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗
Per:Dx64::MEL	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗
Per:Dn64::MEL	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Per:Ix26::MEL	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Per:In26::MEL	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Per:Ix64::MEL	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗*
Per:In64::MEL	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗*
Per:Rx26::MEL	✗	✗	✗	✗	✗	✓	✗	✗	✓	✗	✗	✗
Per:Rn26::MEL	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗
Per:Rx64::MEL	✗	✗	✗	✗	✓	✓	✗	✗	✗	✗	✗	✗
Per:Rn64::MEL	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗
Per:Dx26::SK	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗
Per:Dn26::SK	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗
Per:Dx64::SK	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Per:Dn64::SK	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗
Per:Ix26::SK	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗*
Per:In26::SK	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Per:Ix64::SK	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Per:In64::SK	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Per:Rx26::SK	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Per:Rn26::SK	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Per:Rx64::SK	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Per:Rn64::SK	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Ges::MEL	✗	✗	✗	✗	✗	✗	✓	✓	✓	✗	✗	✗
Ges::NV	✗	✗	✗	✗	✗	✗	✗	✓	✓	✗	✗	✗
Ges::BCC	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Ges::AK	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Ges::BKL	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗
Ges::DF	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗*
Ges::VASC	✗	✗	✗	✗	✓	✓	✗	✗	✗	✗	✗	✗
Ges::SCC	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗

Second, we report the results for the features named in the dermoscopic ABCD rule in Table 4.14. This rule was designed by dermatologists to recognize melanoma. This fact is reflected in the first two columns of Table 4.14. If we compare the models that we trained following Perez et al. (2018) to recognize melanoma and the same

Table 4.14: *Results for the features from the ABCD-rule. For every classifier and feature/test, we indicate that the feature is used with a ✓ or not used with an ✗. We assume a feature is used if the test reports a significant dependence at $p = 0.01$. The models of Perez et al. (2018) are denoted by "Per" followed by their backbone (ResNet-152 (R), Inception-v4 (I), DenseNet-161 (D)) the aggregation method (mean (n), maximum (x)) and the number of augmented samples. Models from the ensemble from Gessert et al. (2020) are denoted by "Ges" followed by the predicted class as melanoma (MEL), melanocytic nevus (NV), basal cell carcinoma (BCC), actinic keratosis (AK), benign keratosis (BKL), dermatofibroma (DF), vascular lesion (VASC) and squamous cell carcinoma (SCC). These results were originally reported in Reimers et al. (2021b)*

Classification	Asymmetry			Border			Color			Derm. Structures		
Model	PC	FCIT	HSIC	PC	FCIT	HSIC	PC	FCIT	HSIC	PC	FCIT	HSIC
Per:Dx26::MEL	✓	✓	✓	✓	✓	✓	✓	✗	✓	✗	✗	✓
Per:Dn26::MEL	✓	✓	✓	✓	✓	✓	✓	✗	✓	✗	✗	✓
Per:Dx64::MEL	✓	✓	✓	✓	✓	✓	✓	✗	✓	✗	✗	✓
Per:Dn64::MEL	✓	✓	✓	✓	✓	✓	✓	✗	✓	✗	✗	✓
Per:Ix26::MEL	✓	✓	✓	✓	✓	✓	✗	✗	✓	✗	✗	✓
Per:In26::MEL	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗
Per:Ix64::MEL	✓	✓	✓	✓	✓	✓	✗	✗	✓	✗	✗	✓
Per:In64::MEL	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗
Per:Rx26::MEL	✓	✓	✓	✓	✓	✓	✗	✗	✓	✗	✗	✓
Per:Rn26::MEL	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	✓
Per:Rx64::MEL	✓	✓	✓	✓	✓	✓	✗	✗	✓	✗	✗	✓
Per:Rn64::MEL	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	✓
Per:Dx26::SK	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✓
Per:Dn26::SK	✗	✗	✗	✗	✗	✓	✗	✗	✗	✓	✗	✓
Per:Dx64::SK	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓
Per:Dn64::SK	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✓
Per:Ix26::SK	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓
Per:In26::SK	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✓
Per:Ix64::SK	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓
Per:In64::SK	✗	✗	✗	✗	✗	✓	✗	✗	✗	✓	✗	✓
Per:Rx26::SK	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓
Per:Rn26::SK	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✓
Per:Rx64::SK	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✓
Per:Rn64::SK	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✓
Ges::MEL	✗	✗	✓	✗	✓	✓	✓	✓	✓	✗	✗	✓
Ges::NV	✗	✗	✓	✓	✗	✓	✓	✓	✓	✗	✗	✓
Ges::BCC	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓
Ges::AK	✗	✗	✓	✓	✗	✓	✗	✗	✗	✗	✗	✓
Ges::BKL	✗	✗	✓	✓	✗	✓	✗	✗	✓	✓	✗	✓
Ges::DF	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Ges::VASC	✗	✓	✗	✗	✓	✓	✗	✗	✗	✗	✗	✓
Ges::SCC	✗	✗	✓	✓	✗	✓	✗	✓	✓	✗	✗	✓

models trained to recognize seborrheic keratosis, all tests agree that all models of the first group use the asymmetry and the border feature while all models of the second group do not. Out of the eight classifiers trained following Gessert et al. (2020), six use the border feature, while none use the asymmetry feature. The color feature is used by almost none of the classifiers. Finally, roughly half of the models trained to

Table 4.15: *Results for the alternative color features. For every classifier and feature/test, we indicate that the feature is used with a ✓ or not used with an ✗. We assume a feature is used if the test reports a significant dependence at $p = 0.01$. The models of Perez et al. (2018) are denoted by "Per" followed by their backbone (ResNet-152 (R), Inception-v4 (I), DenseNet-161 (D)) the aggregation method (mean (n), maximum (x)) and the number of augmented samples. Models from the ensemble from Gessert et al. (2020) are denoted by "Ges" followed by the predicted class as melanoma (MEL), melanocytic nevus (NV), basal cell carcinoma (BCC), actinic keratosis (AK), benign keratosis (BKL), dermatofibroma (DF), vascular lesion (VASC) and squamous cell carcinoma (SCC). These results were originally reported in Reimers et al. (2021b)*

	Color Count			Relaxed C.C.			Variance			Volume		
Classification Model	PC	FCIT	HSIC	PC	FCIT	HSIC	PC	FCIT	HSIC	PC	FCIT	HSIC
Per:Dx26::MEL	✓	✗	✓	✓	✗	✓	✗	✗	✗	✓	✗	✓
Per:Dn26::MEL	✓	✓	✓	✓	✗	✓	✗	✗	✗	✓	✗	✓
Per:Dx64::MEL	✓	✓	✓	✓	✗	✓	✗	✗	✗	✓	✗	✓
Per:Dn64::MEL	✓	✗	✓	✓	✗	✓	✗	✗	✗	✓	✗	✓
Per:Ix26::MEL	✗	✗	✓	✓	✗	✓	✗	✗	✗	✓	✗	✓
Per:In26::MEL	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✓	✓
Per:Ix64::MEL	✗	✗	✓	✓	✗	✓	✗	✗	✗	✓	✗	✓
Per:In64::MEL	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✓
Per:Rx26::MEL	✗	✗	✓	✓	✗	✓	✗	✗	✗	✓	✗	✓
Per:Rn26::MEL	✗	✗	✗	✓	✗	✓	✗	✗	✗	✗	✗	✓
Per:Rx64::MEL	✗	✗	✓	✓	✗	✓	✗	✗	✗	✓	✗	✓
Per:Rn64::MEL	✗	✗	✗	✓	✗	✓	✗	✗	✗	✓	✗	✓
Per:Dx26::SK	✗	✗	✓	✗	✗	✓	✗	✗	✗	✓	✓	✓
Per:Dn26::SK	✗	✗	✗	✗	✗	✓	✗	✗	✓	✓	✓	✓
Per:Dx64::SK	✗	✗	✗	✗	✗	✓	✗	✗	✓	✓	✓	✓
Per:Dn64::SK	✗	✗	✓	✗	✗	✗	✗	✗	✗	✓	✗	✓
Per:Ix26::SK	✗	✗	✗	✗	✗	✓	✗	✗	✗	✓	✗	✓
Per:In26::SK	✗	✗	✗	✗	✓	✓	✗	✓	✓	✓	✓	✓
Per:Ix64::SK	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✓
Per:In64::SK	✗	✗	✗	✓	✗	✓	✗	✗	✗	✓	✗	✓
Per:Rx26::SK	✗	✗	✗	✓	✗	✓	✗	✗	✗	✓	✗	✓
Per:Rn26::SK	✗	✗	✗	✓	✗	✓	✗	✗	✗	✓	✗	✓
Per:Rx64::SK	✗	✗	✗	✓	✗	✓	✗	✗	✗	✓	✗	✓
Per:Rn64::SK	✗	✗	✓	✓	✗	✓	✗	✗	✗	✓	✗	✓
Ges::MEL	✓	✓	✓	✓	✓	✓	✗	✗	✓	✓	✓	✓
Ges::NV	✓	✓	✓	✓	✓	✓	✗	✗	✗	✓	✓	✓
Ges::BCC	✗	✗	✗	✗	✓	✗	✗	✓	✓	✗	✗	✓
Ges::AK	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✓
Ges::BKL	✗	✗	✓	✓	✓	✓	✗	✗	✗	✗	✗	✓
Ges::DF	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Ges::VASC	✗	✗	✗	✗	✓	✓	✓	✓	✓	✗	✗	✗
Ges::SCC	✗	✓	✓	✗	✓	✗	✗	✗	✗	✗	✗	✓

recognize seborrheic keratosis following Perez et al. (2018) and one of the ensemble models trained following Gessert et al. (2020) use the dermoscopic structure feature. We suspect the reason why this feature is used only rarely is the high complexity of the feature. A challenge in the 2018 ISIC workshop, in which contestants were asked to identify areas that showed the dermoscopic structures used in this feature. The task proved to be very difficult with the best submission reaching a Jaccard

Table 4.16: *Results for the bias features. For every classifier and feature/test, we indicate that the feature is used with a ✓ or not used with an ✗. We assume a feature is used if the test reports a significant dependence at $p = 0.01$. The models of Perez et al. (2018) are denoted by "Per" followed by their backbone (ResNet-152 (R), Inception-v4 (I), DenseNet-161 (D)) the aggregation method (mean (n), maximum (x)) and the number of augmented samples. Models from the ensemble from Gessert et al. (2020) are denoted by "Ges" followed by the predicted class as melanoma (MEL), melanocytic nevus (NV), basal cell carcinoma (BCC), actinic keratosis (AK), benign keratosis (BKL), dermatofibroma (DF), vascular lesion (VASC) and squamous cell carcinoma (SCC). These results were originally reported in Reimers et al. (2021b)*

Classification	Age			Sex			Skin Color			Colorful Patches		
Model	PC	FCIT	HSIC	PC	FCIT	HSIC	PC	FCIT	HSIC	PC	FCIT	HSIC
Per:Dx26::MEL	✓	✗	✓	✗	✓	✓	✗	✗	✓	✓	✓	✓
Per:Dn26::MEL	✓	✓	✓	✗	✗	✓	✗	✓	✗	✓	✓	✓
Per:Dx64::MEL	✓	✗	✓	✗	✓	✓	✗	✗	✓	✓	✓	✓
Per:Dn64::MEL	✓	✗	✓	✗	✗	✗	✗	✓	✗	✓	✓	✓
Per:Ix26::MEL	✓	✓	✓	✗	✗	✓	✗	✗	✓	✓	✓	✓
Per:In26::MEL	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓
Per:Ix64::MEL	✓	✓	✓	✗	✗	✓	✗	✗	✓	✓	✓	✓
Per:In64::MEL	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓
Per:Rx26::MEL	✓	✗	✓	✗	✗	✗	✗	✗	✗	✓	✓	✓
Per:Rn26::MEL	✓	✗	✓	✗	✗	✗	✗	✗	✓	✓	✓	✓
Per:Rx64::MEL	✓	✗	✓	✗	✗	✓	✗	✓	✗	✓	✓	✓
Per:Rn64::MEL	✓	✗	✓	✗	✗	✗	✗	✗	✓	✓	✓	✓
Per:Dx26::SK	✓	✗	✓	✗	✗	✗	✓	✗	✓	✓	✓	✓
Per:Dn26::SK	✓	✗	✓	✗	✗	✗	✓	✓	✓	✓	✓	✓
Per:Dx64::SK	✓	✗	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓
Per:Dn64::SK	✓	✗	✓	✗	✗	✓	✓	✓	✓	✓	✓	✓
Per:Ix26::SK	✓	✗	✓	✗	✗	✓	✓	✗	✓	✓	✓	✓
Per:In26::SK	✓	✗	✓	✗	✓	✓	✓	✗	✓	✓	✓	✓
Per:Ix64::SK	✓	✗	✓	✗	✗	✓	✓	✗	✓	✓	✓	✓
Per:In64::SK	✓	✗	✓	✗	✓	✓	✓	✗	✓	✓	✓	✓
Per:Rx26::SK	✓	✗	✓	✗	✗	✓	✓	✗	✓	✓	✓	✓
Per:Rn26::SK	✓	✗	✓	✗	✗	✓	✓	✓	✓	✓	✓	✓
Per:Rx64::SK	✓	✗	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓
Per:Rn64::SK	✓	✗	✓	✗	✗	✓	✓	✓	✓	✓	✓	✓
Ges::MEL	✗	✗	✓	✗	✗	✗	✓	✓	✓	✓	✓	✓
Ges::NV	✓	✗	✓	✗	✗	✓	✓	✓	✓	✗	✓	✗
Ges::BCC	✓	✗	✓	✗	✗	✗	✗	✓	✗	✗	✓	✗
Ges::AK	✓	✗	✓	✗	✗	✓	✗	✓	✓	✗	✓	✗
Ges::BKL	✓	✗	✓	✗	✗	✗	✓	✓	✓	✓	✓	✓
Ges::DF	✗	✗	✗	✗	✗	✗	✗	✓	✓	✗	✓	✓
Ges::VASC	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✓	✓
Ges::SCC	✓	✗	✓	✗	✓	✓	✗	✓	✓	✗	✓	✓

index of 0.307. For more information, we refer the reader to Codella et al. (2019). Finally, only the four classifiers, trained for melanoma recognition following Perez et al. (2018), that use the Densenet-161 as a backbone, as well as three classifiers trained following Gessert et al. (2020) use the color feature. This is surprising since the color is considered one of the most important features in determining the class

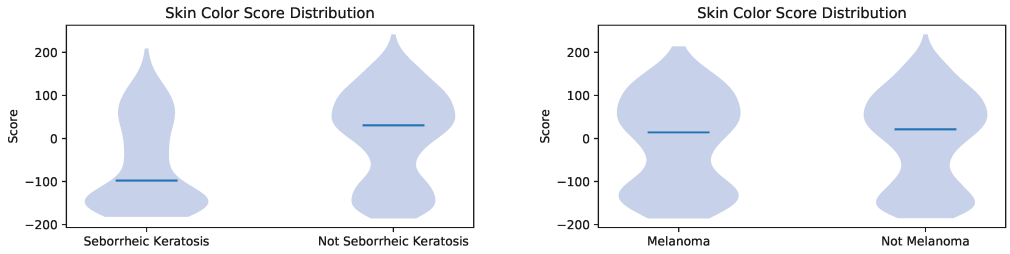
of a skin lesion by dermatologists. To investigate this further, we conduct the next experiment.

Table 4.15 contains the results for the different ways to quantify the color feature. We find that considering a wider window for the colors increases the number of classifiers that use the feature considerably. The relaxed color feature is used by 20 of the 32 models distributed across all three model types. The melanoma models trained following Perez et al. (2018) use the feature in ten out of 12 cases, the seborrheic keratosis models trained in that way and the ensembles following Gessert et al. (2020) use it it half of the cases. In contrast, the variance of the hue is used only by three classifiers. The first is a seborrheic keratosis model trained following Perez et al. (2018) and the two others are models trained following Gessert et al. (2020). Compared to this feature based only on the hue, we consider the volume feature that also considers the pixel's luminosity and saturation. This feature is used by ten of the melanoma and all seborrheic keratosis recognition models trained following Perez et al. (2018). Additionally, two of the ensemble classifiers trained following Gessert et al. (2020) use this feature.

The final experiment in this study considers the known bias features. The results of this experiment are summarized in Table 4.16. Every model included in this study uses at least one of the bias features. To this end, most of the models use the age of the patient and the existence of colorful patches. Both were used by 29 out of the 32 classifiers. All models that did not use at least one of these features are ensemble classifiers trained following Gessert et al. (2020). The remaining two features are less used by the classifiers. The sex of the patient is used by nine out of the 32 models distributed across all three groups of classifiers. For the skin color feature, the results differ among the three groups of classifiers. More than half of the classifiers trained following Gessert et al. (2020) incorporate this feature. In the models trained following Perez et al. (2018), only two out of the twelve melanoma classifiers rely on this feature while, in contrast, all the seborrheic keratosis classifiers trained this way use the skin color.

We find that all seborrheic keratosis models trained following Perez et al. (2018) use the skin color feature. Following up on this observation, we suspected an unknown bias regarding the skin color in the ISIC 2017 training dataset. Figure 4.14 shows the distribution of the feature for both the seborrheic keratosis and the melanoma labels. The median score for seborrheic keratosis images is significantly lower, indicating a clear bias. In contrast, the skin color score of the melanoma and not melanoma images is almost equally distributed.

Another question, we can tackle using these experiments is, which hyperparameters have the most impact on which feature is used by a classifier. The hyperparameters we study here are the following: First, we examine the task for which the network was trained, namely melanoma detection (MEL) or seborrheic keratosis (SK) detection. Second, we compare the backbone classifier, namely the DenseNet-161 (D), the ResNet-152 (R), and the Inception-v4 (I). The third hyperparameter is the aggregation method, which can be the mean (n) or the maximum (x), and the last hyperparameter is the number of samples during the test time augmentation,



(a) (b)

Figure 4.14: *The distributions of the skin color score in the ISIC 2017 training dataset (Codella et al., 2018) split according to the seborrheic keratosis and melanoma labels. The difference in Figure 4.14a indicates a clear bias.*

namely 26 and 64. If a hyperparameter is important, it is suitable to predict whether a certain feature is used. This means that the use of a feature is more homogeneous when fixing the hyperparameter. To this end, we measure the homogeneity of a feature's usage given a hyperparameter. We focus on the models trained following Perez et al. (2018) for this experiment.

To quantify the agreement, we consider the distribution of outcomes, "feature used" and "feature not used," given a hyperparameter X and calculate its entropy,

$$H(X) = -(p_{\checkmark} \log_2 p_{\checkmark} + p_{\text{✗}} \log_2 p_{\text{✗}}). \tag{4.104}$$

Here, $p_{\checkmark}$ is the probability of the feature being used and $p_{\text{✗}}$ is the probability of the feature not being used calculated from the frequencies in tables 4.13 - 4.16 of all tests on a classifier that uses hyperparameter X. The entropy is equal to one in case $p_{\checkmark} = p_{\text{✗}}$ and equal to zero in case $|p_{\checkmark} - p_{\text{✗}}| = 1$. Since the entropy achieves its minimum of zero if all results for a fixed hyperparameter agree and we want a high value to indicate importance, we use $1 - H$ instead.

The results can be found in Table 4.17. The most important influence is, as expected, the class that the model is trained to recognize. The second most important hyperparameter is the backbone architecture of the classifier followed by the aggregation method and the number of samples in that order. However, the standard error shows, that these results only give use a qualitative idea and more research is needed to make a definite statement.

4.3.3 Experiments on Debiasing

In this section, we present the experiments that we conducted to investigate the performance of our new adversarial debiasing method, which we introduced in Section 4.2.2. To this end, we present three experiments. For the first experiment, in Section 4.3.3.1 we created a synthetic dataset that maximizes the difference between the conditional and the unconditional criterion. To achieve this, we maximize the dependence between the signal S and the bias variable B. While we find strong results for all our implementations, as expected, the methods from the literature are

Table 4.17: *Agreement scores and standard error for different hyperparameters of the models trained following Perez et al. (2018). The possible tasks are melanoma and seborrheic keratosis recognition. The three backbones are DenseNet-161, Inception-v4 and ResNet-152. We employ mean and maximum aggregation and use either 26 or 64 augmented examples*

Hyperparameter	Score
Task	0.774 ± 0.066
Backbone	0.595 ± 0.107
Aggregation method	0.563 ± 0.112
Number of examples	0.549 ± 0.113

not able to outperform the baseline. To ensure that this difference in performance originates from the change from the unconditional to the conditional dependence criterion and not from other changes in the implementation, we present the results of an ablation study in Section 4.3.3.2. Finally, in Section 4.3.3.3, we show that this effect is also observable on real-world images. To this end, we created eleven biased training sets out of the dataset of cats and dogs introduced by Lakkaraju et al. (2016). On these test sets, we can not only show that the implementation of our models outperforms all unconditional methods from the literature, but we, furthermore, observe a strong correlation between the strength of the bias and improvement of our method over its unconditional counterpart. The three experiments described in this section were originally included in Reimers et al. (2021a).

To evaluate our experiments, we measure the accuracy on an unbiased test set. This is not the only possibility to evaluate debiasing methods. Other common evaluation methods include the "equalized odds" (Hardt et al., 2016),

$$(R \perp\!\!\!\perp B \mid L), \tag{4.105}$$

or "demographic parity" (Dwork et al., 2012),

$$(R \perp\!\!\!\perp B), \tag{4.106}$$

both of which are the same in this situation since the test set is unbiased. The main drawback of these measures is that they are binary and, hence, rather coarse-grained. We focus on the accuracy on unbiased test sets in this paper. The reason is that we designed this method for situations in which a dataset is biased, but we expect the system to be used in an unbiased, real-world situation. Hence, the accuracy on an unbiased test set is our goal, and evaluating it directly is the most precise quality measure for our method.

4.3.3.1 Experiments on Synthetic Data

To evaluate the quality of our suggested adversarial debiasing method, we, first, conduct an experiment on synthetic data. To this end, we build a small dataset

of tiny eight-by-eight pixel images. All but twenty pixels in these images are dark. The twenty light pixels have two features. The first feature is the shape of the light pixels, that form either a cross or a square. The second feature is the color of the light pixels which is either green or violet.

More specifically, we create the images in the HSV color space. Depending on the two signals, we first determine the value of each pixel. Since the first signal contains the shape of the high-intensity pixels, we set the value of pixels within the shape to 0.8 and of pixels outside of the shape to zero. Then independent identically distributed noise following a uniform distribution on the interval $[0, 0.2]$ is added to the intensity of all pixels. To add the second signal we use the hue of the pixels. To this end, the hue of all pixels is either set to 0.3 (green) or 0.9 (violet). We add noise to the hue of every pixel. The noise is also sampled from a uniform distribution but this time on the interval $[-0.1, 0.1]$. In the end, we convert the images into the RGB color space, resulting in a nonlinear mixing of the two signals. Example images of this dataset can be found in Figure 4.15.

The two criteria on whether a feature is used by a deep neural network, the dependence criterion in (4.66) and the conditional dependence criterion (4.65) agree if the bias variable B is independent of the label L, meaning that the dataset is unbiased

$$B \perp\!\!\!\perp L \quad \Rightarrow \quad (B \perp\!\!\!\perp R \quad \Leftrightarrow \quad B \perp\!\!\!\perp R \,|\, L)\,. \tag{4.107}$$

Further, the weaker the bias in the dataset, the closer are the two criteria. Since we want to compare the effect of the two criteria for adversarial debiasing, we aim to maximize the difference between these two criteria. Consequently, we maximize the bias in this synthetic dataset. In our training set, all 600 images of squares are violet and all images of crosses are green. In contrast, In the test set these two signals are independent, meaning it contains as many green as violet crosses and as many green as violet squares. Example images from the train and test set can be found in Figure 4.15.

For two reasons, we limit the training set to this small number of images. First, since the task is relatively easy, only a few images are needed to solve it and we do not want to create an unrealistic setting in which the network has a much easier task to extract a specific feature than it would have in a realistic setting. Second, the use-case for adversarial debiasing is small datasets, since for large datasets other methods, such as creating synthetic examples or downsampling the dataset can be more advantageous.

Many authors argue that neural networks prefer features that are related to the color of an object over features that concern the shape of an object or vice versa. Since one of our features considers the color and one the shape, we have to be careful that these preferences do not influence our investigation. To this end, we use two setups: In the first setup, the shape of the high-value pixels is the signal S. From this signal, we derive a binary label L which is either “cross” or “square”. As the bias variable B, we use the color signal. We denote this setup as Setup I. For the second setup, we invert the roles of the color and shape signals. The signal S is given by the color and determines the binary label L which is either “green” or “violet”.

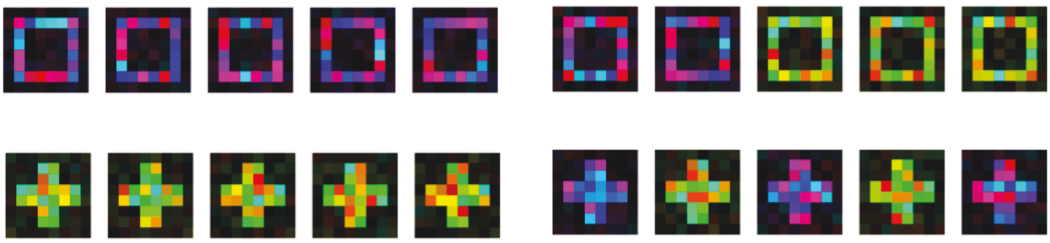


Figure 4.15: *Example images from the synthetic dataset. In the training set, the color and the shape are dependent. Every image of a cross is green and every image of a square is violet. In the test set, the two signals are independent. It contains as many violet as green crosses and as many violet as green squares. This figure was originally published in Reimers et al. (2021a).*

The shape determines the bias feature B. We score the color feature by calculating the mean hue over all pixels in the image. To score the shape feature, we consider the subset of pixels that are high valued in the cross shape but are low-valued in the square shape and the subset of pixels that are high-valued in the square shape but low valued in the cross shape. The feature B is then scored as the mean value of the pixels in the former group minus the mean value of pixels in the latter group. We denote this setup by Setup II.

As the classifier in these setups, we use a convolutional neural network. The first two layers of this neural network are convolutional layers with 16 filters, each of size three-by-three and ReLU activations. Following these convolutional layers, the network uses two dense layers, one with 128 neurons and ReLU activation and one with two neurons and softmax activation. This classifier is used on its own as a baseline, and as the backbone to each method, the methods presented in this work, as well as methods we took from the literature.

We compare the method presented in this study with four method from the literature which we reimplemented following their respective publications. These four methods are the method presented by Adeli et al. (2021) which penalizes the correlation between the bias variable B and the representation transformed by a continuous function $f(P)$. We parametrize the function by a deep neural network. The second method was suggested by Zhang et al. (2018a), which penalizes the predictability of the bias variable B from the representation P. As we demonstrated in Section 4.2.2, the predictability and the correlation mentioned above are related. Since the third method is also proposed in Zhang et al. (2018a), we denote this first method by Zhang et al. (2018a) I and the following method by Zhang et al. (2018a) II. For this second method proposed by Zhang et al. (2018a) the authors penalize the predictability of the bias variable B from the representation P and the true label L. Finally, we reproduced a method of Kim et al. (2019). This method uses two strategies. First, it maximizes the predictability of the bias variable B measured by the mean square error, second, it maximizes the entropy of the distribution over different values for the bias variable given the representation B.

Some of the methods from the literature require the training of additional neural networks. Whenever a method, including the variant of our method that uses the maximum partial correlation coefficient, requires the training of an additional neural network, we use a neural network with one fully connected, hidden layer of size 1024 and ReLU activations. If the respective publications of the literature methods do not specify differently, we use the Adam optimizer (Kingma and Ba, 2014) to optimize the weights of the deep neural networks.

We run the experiment for each method, the four literature methods and the three implementations suggested in Section 4.2.2.3, 100 times for each of the two setups explained above. All hyperparameters are optimized for the two setups independently. We restrict the hyperparameters for each of the literature methods to the values described in the respective publications and used hyperparameter optimization only for the values which are either problem-dependent or not specified in the publications. We report the average test set accuracy and the standard error in Table 4.19.

If a method would decide half of the example based on the shape and the other half based on the color, it would reach an accuracy of 0.75 in each setup. In our results, we find that the baseline reaches an accuracy above this threshold in both setups. This demonstrates that the hyperparameters influence which features (color or shape) is extracted and used by a deep neural network. This strong influence of hyperparameter might obscure the influence of the adversarial debiasing methods. Hence, it is very important, to use rigorous hyperparameter optimization. We used a grid search where we train ten models for each hyperparameter configuration and used the hyperparameter optimization with the highest mean validation set accuracy. The list of hyperparameters can be found in Table 4.18. For each method, we report the learning rate of the backbone classifier (lr_c), the number of epochs (Nr. Ep.), and, for all methods other than the baseline, the weight of the adversarial debiasing loss β. For each method that requires a second neural network to calculate the debiasing loss, we report the learning rate of these additional neural networks by lr_b. Finally, we report whether we forced the mini-batches used during training to be balanced in the classes (Bal.).

All methods from the literature employ the unconditional independence criterion (4.66). With one exception, none of them can improve upon the baseline. This is what we expected from our theoretical investigations in Section 4.2.2.2 and the literature. When trained with an unconditional independence criterion, the methods trade biasedness for test set accuracy. Especially in a dataset with a very strong bias, such as the one presented here, this effect has a huge influence. In contrast, all implementations of our method outperform the baseline and consequently all methods from the literature. In the first setup, the variance in test set accuracy among the implementations of our method using the new conditional dependence criterion (4.65) is 0.19 percentage points. Similarly, the variance among the test set accuracy of the literature methods using the unconditional independence criterion (4.66) is 0.53 percentage points. In contrast to these variances within these two groups, the difference between the worst implementation of our method and the

Table 4.18: *The hyperparameters for all methods and all experiments*

Method	lr_c	Nr. Ep.	β	lr_b	Bal.
			Setup I		
Baseline	3e−5	30	–	–	False
Adeli et al. Adeli et al. (2021)	3e−4	1000	1	3e−4	False
Zhang et al. I Zhang et al. (2018a)	3e−3	30	2	3e−3	False
Zhang et al. II Zhang et al. (2018a)	3e−3	30	0.5	3e−3	False
Kim et al. Kim et al. (2019)	3e−4	1000	1	3e−4	False
Ours(CMI)	1e−5	100	0.0625	1e−5	False
Ours(MPCC)	3e−5	30	0.0625	3e−5	False
Ours(HSCONIC)	3e−5	30	0.0625	3e−5	True
Unconditional HSIC	3e−5	30	0.003	3e−5	True
Unconditional MI	1e−5	100	0.0625	1e−5	False
Ours(MPCC) – only MCC	3e−4	1000	1	3e−4	False
Ours(MPCC) – only PC	3e−5	30	0.0625	3e−5	False
			Setup II		
Baseline	3e−5	100	–	–	False
Adeli et al. Adeli et al. (2021)	3e−5	100	1	3e−5	False
Zhang et al. I Zhang et al. (2018a)	3e−3	30	0.5	3e−3	False
Zhang et al. II Zhang et al. (2018a)	3e−3	30	0.5	3e−3	False
Kim et al. Kim et al. (2019)	3e−5	100	1	3e−5	False
Ours(CMI)	1e−5	100	0.05	1e−5	False
Ours(MPCC)	3e−5	30	0.0625	3e−5	False
Ours(HSCONIC)	3e−5	30	0.0625	3e−5	True
Unconditional HSIC	3e−5	30	0.0625	3e−5	True
Unconditional MI	1e−5	100	0.05	1e−5	False
Ours(MPCC) – only MCC	3e−5	30	0.0625	3e−5	False
Ours(MPCC) – only PC	3e−5	30	0.0625	3e−5	False
			Real-World Data		
Baseline	1e−2	150	–	–	False
Adeli et al. Adeli et al. (2021)	1e−2	150	1	3e−4	False
Zhang et al. I Zhang et al. (2018a)	1e−2	150	1	3e−5	False
Zhang et al. II Zhang et al. (2018a)	1e−2	150	1	3e−5	False
Ours(HSCONIC)	1e−2	150	1	1e−5	False
Unconditional HSIC	1e−2	150	1	1e−5	False

Table 4.19: *The results from 100 experimental runs for our method and all baseline methods. For both experiments, we report the mean accuracy ± standard error. The best results are marked in **bold**. These results were first published in Reimers et al. (2021a)*

Method	Setup I	Setup II
Baseline	0.819 ±0.016	0.791 ±0.016
Adeli et al. (2021)	0.747 ±0.015	0.776 ±0.014
Zhang et al. (2018a) I	0.736 ±0.018	0.837 ±0.017
Zhang et al. (2018a) II	0.747 ±0.016	0.750 ±0.013
Kim et al. (2019)	0.771 ±0.012	0.767 ±0.016
Ours(CMI)	0.840 ±0.014	**0.871** ±0.012
Ours(HSCONIC)	0.846 ±0.021	0.868 ±0.013
Ours(MPCC)	**0.854** ±0.013	0.867 ±0.013

best method from the literature is 6.9 percentage points. In Setup II, the numbers are similar, only one of the methods suggested by Zhang et al. (2018a) outperforms the baseline. In this setup, the variance among the implementations of our methods is smaller than 0.01 percentage points. The variance among the test accuracy of the literature methods is 0.43 percentage points. The difference between the worst implementation of our method and the best performing method from the literature is 3.0 percentage points.

We summarize the findings from these experiments in the following two observations. The synthetic dataset is suitable to demonstrate the difference between the two debiasing criteria. As intended, it proves challenging for the methods from the literature. In the first setup, none of the literature methods can outperform the baseline and in Setup II, only one of the methods can outperform the baseline. This is expected since it is a known fact, that these methods trade unbiasedness for accuracy and fail for strong biases. The reason for this tradeoff is explained in Section 4.2.2.2. In contrast, all implementations of our new conditional independence criterion for debiasing reach higher test set accuracy than the baseline and, consequently a higher test set accuracy than all methods from the literature.

4.3.3.2 Ablation Study

In the experiment on synthetic data that we presented in Section 4.3.3.1, we found that the proposed implementations outperformed the methods from the literature. To demonstrate that this increase in performance originates from our new conditional independence criterion (4.65) and not, for example, from details in the implementation, we present an ablation study in this section. In this study, we compare the implementations we proposed for the conditional dependence criterion with counterparts that are implemented in the same way but use the unconditional dependence test. We compare eight variations of the implementations described in Section 4.2.2.3. First, for the method based on conditional mutual information

Table 4.20: *The results of the ablation study. Every method is trained on a biased training set and evaluated on an unbiased test set according to the indicated setup. We report the accuracy averaged over 100 runs and the standard error. The best results are marked in* ***bold****. This table was originally included in Reimers et al. (2021a)*

Method	Setup I	Setup II
Ours – CMI	0.583 ± 0.010	0.833 ± 0.011
Ours – MI	**0.840** ± 0.014	**0.871** ± 0.012
Ours – HSCONIC	0.744 ± 0.011	0.590 ± 0.011
Ours – HSIC	**0.846** ± 0.021	**0.868** ± 0.013
Adeli et al. (2021)	0.747 ± 0.015	0.776 ± 0.014
Ours(MPCC) – only MCC	0.757 ± 0.016	0.807 ± 0.015
Ours(MPCC) – only PC	0.836 ± 0.014	0.830 ± 0.015
Ours(MPCC) – complete	**0.854** ± 0.013	**0.867** ± 0.013

("Ours – CMI"), we create an implementation that is equal in every way other than the fact that it uses unconditional mutual information. We denote this method as "Ours – MI". Second, we created an implementation that is in every way equal to our method that employs the Hilbert-Schmidt conditional independence criterion ("Ours – HSCONIC") other than that it uses the unconditional Hilbert-Schmidt independence criterion. We denote this method as "Ours – HSIC". Finally, since we made two individual changes to turn the predictability criterion suggested by Adeli et al. (2021) and Zhang et al. (2018a) into our maximum partial correlation criterion we compare the four different loss functions described in Section 4.2.2.3. First, in Table 4.20, we denote the predictability criterion following Adeli et al. (2021), by Adeli et al. (2021). The method that improves upon this method by using the maximum correlation coefficient instead of the predictability criterion as explained in Equation (4.95) is denoted by "Ours(MPCC) – only MCC". The method that improves upon Adeli et al. (2021) by using the partial correlation instead of the correlation,

$$\text{MPCC}_{PConly}(R, B \mid L) = \max_{f} \text{PC}(f(R), B \mid L), \tag{4.108}$$

is denoted by "Ours(MPCC) – only PC". Finally, our implementation using the maximum partial correlation criterium is denoted by "Ours(MPCC) – complete".

In both setups, we find that the implementations using the conditional independence criteria outperform their unconditional counterparts for all three implementations. For the implementations based on mutual information and conditional mutual information as well as the implementations based on the Hilbert-Schmidt independence criterion and the Hilbert-Schmidt conditional independence criterion, the accuracy increased on average by 0.169. For the implementations related to the maximum correlation criterion, we evaluate the effects of the two different changes we made individually. The first change, from the predictability to the maximum

correlation coefficient, increased the test set accuracy on average by 2 percentage points in the unconditional case (Adeli et al. (2021) vs. "Ours(MPCC) – only MCC") and on average by 2.8 percentage points in the conditional case ("Ours(MPCC) – only PC" vs. "Ours(MPCC) – complete"). In contrast, the second change from unconditional to the conditional criterion increased the test set accuracy on average by 7.2 percentage points when using the predictability criterion (Adeli et al. (2021) vs. "Ours(MPCC) – only PC") and on average by 7.9 percentage points when using the maximum correlation criterion ("Ours(MPCC) – only MCC" vs. "Ours(MPCC) – complete"). We conclude, that the effects that we observed in Section 4.3.3.1 can be attributed to the change from the unconditional to the conditional dependence criterion.

Even further, since our unconditional methods perform worse than the comparable methods from the literature, we conclude that we could improve our results further by improving details in the implementations.

4.3.3.3 Real-world data

Table 4.21: *Experimental results on the cats and dogs dataset introduced by Kim et al. (2019). All methods were trained on a dataset in which p% of all dogs are dark-furred dogs and p% of all cats are light-furred. The first column of the table indicates the fraction p. The following columns contain the accuracies on an unbiased test set averaged over three runs and the standard error. We marked the best results in **bold**. The results in this table were originally presented in Reimers et al. (2021a)*

Fra.	Baseline	Adeli et al. (2021)	Zhang et al. (2018a) I	Zhang et al. (2018a) II	Ours (HSIC)	Ours (HSCONIC)
0%	**0.627** ± 0.004	0.597 ± 0.004	0.590 ± 0.002	0.617 ± 0.001	0.611 ± 0.003	0.615 ± 0.005
10%	0.800 ± 0.001	0.774 ± 0.002	0.779 ± 0.005	0.785 ± 0.007	0.759 ± 0.012	**0.801** ± 0.001
20%	0.845 ± 0.003	0.829 ± 0.000	0.812 ± 0.002	0.809 ± 0.005	0.816 ± 0.002	**0.855** ± 0.004
30%	0.852 ± 0.007	0.842 ± 0.003	0.837 ± 0.004	0.834 ± 0.003	0.834 ± 0.002	**0.863** ± 0.002
40%	0.859 ± 0.007	0.855 ± 0.004	0.870 ± 0.002	0.850 ± 0.001	0.861 ± 0.003	**0.875** ± 0.003
50%	0.859 ± 0.006	**0.866** ± 0.003	0.856 ± 0.001	0.853 ± 0.001	0.863 ± 0.004	0.860 ± 0.002
60%	**0.866** ± 0.006	0.837 ± 0.001	0.850 ± 0.003	0.860 ± 0.004	0.844 ± 0.001	0.856 ± 0.005
70%	0.844 ± 0.003	0.854 ± 0.003	0.835 ± 0.005	0.841 ± 0.005	0.835 ± 0.003	**0.859** ± 0.000
80%	0.829 ± 0.002	0.822 ± 0.005	0.820 ± 0.005	0.826 ± 0.003	0.820 ± 0.007	**0.836** ± 0.002
90%	0.773 ± 0.010	0.743 ± 0.001	0.758 ± 0.001	0.731 ± 0.002	0.757 ± 0.003	**0.791** ± 0.004
100%	0.612 ± 0.001	0.612 ± 0.004	0.604 ± 0.001	0.609 ± 0.001	0.606 ± 0.002	**0.616** ± 0.002

In Section 4.3.3.1, we have demonstrated the effectiveness of our approach on a synthetic dataset. Further, in Section 4.3.3.2, we demonstrated that the observed improvements over the methods from the literature can be attributed to the change from the unconditional to the conditional independence criterion. In this section, we demonstrate that this improvement generalizes to real-world images. To this end, we use a dataset of cats and dogs that was introduced by Lakkaraju et al. (2016) for the same purpose. In this dataset, each image has two different labels. The first label indicates whether the image contains a cat or a dog and the second label indicates whether the animal in the image is light-furred or dark-furred. These

labels are manually created. The fact that each image has two labels allows us to create biased training sets and an unbiased test set. To create these sets, we first divide the dataset into four groups, namely the light-furred dogs, the dark-furred dogs, the light-furred cats and the dark-furred cats. To create the test set, we then select twenty percent of the smallest of these four sets and the same number of images from the other three sets. This creates an unbiased test set, in which the fur color has no information on whether the animal in the image is a cat or a dog. Using the remaining images, we create eleven training sets. To this end, we start with a training set that contains only light-furred dogs and dark-furred cats. In Table 4.21 this dataset is denoted as 0% because the training set contains zero percent dark-furred dogs. We continue creating datasets with a higher fraction of dark-furred dogs and light-furred cats. In Table 4.21 a fraction of $p\%$ denotes that in the training set a fraction of $p\%$ of all images of dogs contain dark-furred dogs while the other $(1-p)\%$ images of dogs contain light-furred dogs. At the same time, this means that of all images of cats in the dataset, a fraction of $p\%$ is light-furred and a fraction of $(1-p)\%$ is dark-furred. All of these datasets contain as many images of dogs as they contain images of cats. To further guarantee a fair evaluation, we created all of the datasets to have the same number of images in total. Consequently, the number of images in the training set is limited by two times 80% of the smallest fur color-class combination, as it will have to provide half of the images in the $p = 0\%$ or the $p = 100\%$ case. Unfortunately, this limits us to training sets of 2469 images, which is 14.7% of the training data of the original dataset.

In Table 4.21, we report the test set accuracy and the standard error averaged over three runs for six methods. The first method is a baseline, a neural network without any debiasing method. Further, we report results for the method presented by Adeli et al. (2021) and both methods presented in Zhang et al. (2018a). Finally, we report the results for one of our implementations, namely the implementation based on the Hilbert-Schmidt independence criterion "Ours (HSIC)" and the implementation based on the Hilbert-Schmidt conditional independence criterion "Ours (HSCONIC)". For details see Section 4.3.3.1 and Section 4.3.3.2.

As the backbone network for the classifiers in this experiment on real-world data, we use a ResNet-18 proposed by He et al. (2016). The network is trained for 150 epochs using the Adam optimizer proposed by Kingma and Ba (2014). The learning rate follows a cosine decay with warm restarts (Loshchilov and Hutter, 2016). Furthermore, we use random cropping during training and center cropping during inference (Simonyan and Zisserman, 2014) as well as a progressive resizing scheme. As in the previous experiments, whenever a method requires an additional neural network, we use a network with one hidden layer of 1024 neurons. To optimize the hyperparameters for the baseline, we apply a grid search. For the hyperparameters of literature methods and our implementations we adapt the hyperparameters of Setup I in Section 4.3.3.3 by comparing the hyperparameters of the baselines in Setup I of Section 4.3.3.3 and the baseline in this experiment.

The test set accuracies that we reached in this study are lower than test set accuracies reported in other studies, for example, Kim et al. (2019). This discrepancy is

due to two factors. First, we use only 14.7% of the training data, while other works use all of the data for training. Second, other work uses pre-training on ImageNet (Russakovsky et al., 2015). However, the ImageNet dataset already contains several thousand images of cats and dogs. Hence, pre-training on this dataset can influence whether the neural network picks up the bias from the dataset. Since the applications of debiasing are more likely in situations where no large unbiased datasets for pretraining exist, we choose to refrain from pretraining on ImageNet in our experiments.

We find that the method suggested in this work outperforms the baseline and the methods from the literature. Since both labels, the "cat"/"dog" as well as the "dark-furred"/"light-furred", are binary, the two signals cannot be distinguished if either all dogs are dark-furred or all dogs are light-furred. No adversarial debiasing method can work in this situation. Further, if exactly half of the cats and half of the dogs are light-furred and the other half is dark-furred the signals are independent. Hence, in this situation no adversarial debiasing is needed.

This leaves eight proper adversarial debiasing scenarios among the scenarios listed in Table 4.21. Out of these eight scenarios, our method performs best in seven. Furthermore, it reaches the highest overall accuracy of 0.875. To determine the significance of the differences between our methods and the methods from the literature, we use the standard error to determine the 95% confidence interval. In six out of the seven scenarios where our method performs best, no other method performs within this 95% confidence interval. As expected, the methods from the literature, as well as our unconditional implementation of the Hilbert-Schmidt independence criterion only improve upon the baseline in the scenarios with little bias. While in situations with a strong bias, the methods decrease the test set accuracy of the baseline.

Similar to the ablation study in Section 4.3.3.2, we compare the implementation based on the Hilbert-Schmidt conditional independence criterion ("Ours (HSCONIC)") and the implementation based on the unconditional Hilbert-Schmidt independence criterion ("Ours (HSIC)"). We find that the conditional implementation outperforms the unconditional one in all but the unbiased scenario. More importantly, the differences between the conditional and unconditional implementations is larger if the dependence between the two signals is stronger. We find that the correlation between the absolute value of the difference between the fraction of light-furred and dark-furred dogs and the difference between the conditional and unconditional implementations is 0.858.

Further, we compare a baseline classifier and a classifier using HSCONIC both trained on a training set with 70% dark-furred dogs. As expected, 65% of all images correctly classified by the HSCONIC classifier but not by the baseline classifier are light-furred dogs or dark-furred cats. In contrast, roughly half (56%) of the images correctly classified by the baseline classifier but not by the classifier using HSCONIC are light-furred dogs or dark-furred cats. We display 16 random images from the test set that are correctly classified by the HSCONIC classifier but not by the baseline classifier in Figure 4.16.

Figure 4.16: *Images correctly classified by the classifier using HSCONIC but not the baseline both trained on the dataset with* 70% *dark-furred dogs. Most of the images are dogs with light fur or cats with dark fur. Even the dog in the third column of the first row, which is labeled as dark-furred, is very light for a dog with dark fur.*

5 | Conclusions

In this work, we presented multiple steps towards understanding and debiasing deep neural networks. In this section, we summarise these steps and draw conclusions. We start by describing the conclusions from our investigations into the definition and the reason for adversarial examples, which we presented in Section 3. We describe these conclusions in Section 5.1. The main result of this work is the method to determine whether a feature is used by a deep neural network, which we described in Section 4. Hence, in Section 5.2, we summarize our conclusions about this main method. To demonstrate that this method is applicable to real-world tasks and returns reasonable results, we discussed two applications of this method. These applications are the investigation into automatic skin lesion classifiers presented in Section 4.2.1 and the use of the method for adversarial debiasing, which we presented in Section 4.2.2. We summarize the findings for the former in Section 5.2. This includes findings that demonstrate the usefulness and correctness of the main contribution in Section 4.1 as well as the results that concern the specific task of automatic skin lesion classification. The same is true for the results of the adversarial debiasing study described in Section 4.2.2. In that study, we reached results to corroborate the fact that our method can correctly identify whether a feature is utilized by a deep neural network as well as results that only concern its use as an adversarial debiasing method. We summarize the results for the latter in Section 5.4.

5.1 Adversarial Examples

In the work described in Section 3, we took a step towards linking the properties of datasets to adversarial robustness. We focused on two defining properties of adversarial examples, namely the size of the perturbation and the non-resemblance to the target class. Both of these properties depend on the dataset and are difficult to compare between data sets.

For the first defining property, we introduced τ^0, which makes the notation of *small* independent of the data set. For the second defining property, we introduced $\mathfrak{R}$ to measure the resemblance. While we think that both of these tools will be useful in future research, some problems still exist. For example, in big data sets, both of these quantities are computationally expensive to calculate, and we have to rely on evaluating them on subsets. However, since our goal is to understand how

properties of the data set influence the adversarial robustness, most of the time we have to rely on small, well-understood data sets and can not employ the big data sets for which many properties are hard to determine.

Further, we show that our methods can be used to establish links between properties of the data set and properties of adversarial examples. We demonstrated this for the multimodality of the distribution of examples from one class.

5.2 Identifying Relevant Features

In Section 4 of this work, we presented a novel method to determine whether a feature is considered relevant by a deep neural network. Our method is based on the framework of causal inference, which we explained in Section 2.1. Our method has several key properties that tell it apart from other methods and outline the use case for our method.

The first property is that our method, in contrast to, for example, saliency maps (c.f. Section 2.4.1), evaluates the global behavior of the classifier represented by the deep neural network and not the local behavior at one example.

Second, our method can be applied post hoc to a black-box classifier. We do not need access to any intermediate representations or gradients and we do not need to alter the training process in any way. This is a big advantage, as it allows our method to be used for pre-trained deep neural networks and even any supervised classifier that is not a deep neural network. This includes other supervised learning algorithms as, for example, kernel-SVMs but also new supervised learning algorithms that we do not know yet.

Third, our method is built on the strong theoretical background of causal inference. This is very important because, as described in Section 2.4, verifying methods that decide whether a feature is used by a deep neural network empirically is difficult. Consequently, we need a strong theoretical foundation to ensure, that the predictions of our method are correct and that it is not vulnerable to, for example, confounding.

However, most importantly, our method is suited to evaluate whether features are used that are not represented by areas of the input image, but are descriptions of the whole image. Examples of these kinds of features are the relative position of an object in an image, which we used in Section 4.3.1, the symmetry, length of the border, the color variation within the skin lesion, which we used in Section 4.3.2, and bias features such as the age or sex of patients wich we used in Section 4.3.2. Furthermore, this property allows us to differentiate between features that are represented by the same region of the input image. One example of such a feature, which we presented in Section 4.3.1, is the two features that describe different properties of the yellow lores that distinguish between white-crowned sparrows (Zonotrichia leucophrys) and white-throated sparrows (Zonotrichia albicollis). In addition, we explained in Section 4.3.1 how our method can be used to compare classifiers with respect to specific properties if we do not have access to a specialized dataset that contains images that only differ in this property. This is especially useful

in applications such as medical image classification tasks or earth system science, where it might be expensive, unethical or impossible to sample specific inputs.

As a result of these properties, we believe that our method has clear use cases for certain scientific questions. One of these use cases is medical images, as we outline in Section 4.2.1. We believe that in those tasks, our method is a valuable addition that can provide additional and different insights than other methods, for example, saliency maps.

In Section 4.1.5 we explain, that our method is suitable for many algorithms even if these algorithms do not fit the original description. Further, in Section 4.3.1 we demonstrated that the method returns correct results for simple classifiers and, in Section 4.3.2, we demonstrated that it returns at least reasonable results even in the complex real-world situation of automatic skin lesion classification.

However, we also found clear limits and drawbacks of our method. First, our method returns a binary decision on whether a feature is used or not. Neither can we use it to understand what the influence of a feature is nor can we use it to find whether a feature is more important than another. It is not suitable to compare different features and should only be used for single features.

Second, our method can not differentiate between features that contain the same information. This is a limitation because semantic features often contain mutual information. In these situations, our method might indicate that a specific feature is used, even though, the actual feature that is used is a different feature that just contains a lot of mutual information. An example of such a pair of features could be the surface and the volume of an object.

Third, while we reduce the semantic question of whether a feature is extracted and used by a deep neural network to the more objective decision of a conditional statistical dependence test, conditional statistical dependence testing is not trivial. Each dependence test has advantages and disadvantages. Many dependence tests, furthermore, contain hyperparameters, which have a strong influence on the behavior of the dependence test. Hence, in each application of our methods, one has to select a dependence test, including a set of suitable hyperparameters for this dependence test, that fit the properties and connections of the dataset. Since these properties and, in particular, the connections in the dataset are often unknown, this remains a difficult problem.

To summarise, through the theoretical considerations in Section 4.1 and the various experiments in Section 4.3, we show that our method is applicable to many situations of supervised learning and that it returns reasonable results for deep neural network classifiers on real-live datasets, such as HAM10000 (Tschandl et al., 2018), CUB200 (Welinder et al., 2010) and MS COCO (Lin et al., 2014). We can use it to evaluate classifiers, understand classifiers better and further the trust into automatic classifiers.

5.3 An Application to Skin Lesion Classification

In the previous section, we summarized our general conclusions on the method discussed in this work. However, as a use case and to illustrate the application of the method further, in Section 4.2.1, we applied the method to an example task. This example is the task of automatic skin lesion classification. Our method is suitable for this task because the number of images is limited and it is difficult to collect more data. Further, the datasets contain multiple known biases as well as features that are known to be relevant to the classification of skin lesions. Finally, the task is difficult enough to represent state-of-the-art deep learning, and understanding automatic classifiers is especially relevant in mecial tasks like this one.

Consequently, we divide the conclusions from this section into two parts. First, we summarize our findings towards the general method presented in this work, and, afterward, we describe the results of our findings, that focus on the application of automatic skin lesion classification.

Towards the former, we find that the method demonstrated in this work produces meaningful results. It does not, or very rarely, consider a feature that contains little to no information on the class of the skin lesion as relevant. This demonstrates that the method is suitable for complex real-world classification tasks, such as automatic skin lesion classification, especially because the features for this validation are similar in complexity to the relevant features that we evaluate later. Furthermore, one of the most significant observations is that the models trained for melanoma recognition following Perez et al. (2018) use the asymmetry and border features, while the models trained for seborrheic keratosis recognition following the same work, do not. This is expected because the dermoscopic ABCD rule was developed to distinguish melanoma and nevi and not to recognize seborrheic keratosis. This study underpins the fact that our method is applicable to a wider range of features than, for example, saliency maps. Saliency maps could only work for the colorful patches feature out of the 15 features analyzed in this work.

Towards the latter, we investigate four questions: First, we evaluate, whether the automatic, state-of-the-art deep neural network classifiers use the medically relevant features named in the dermoscopic ABCD rule. Second, we check if these classifiers also base their decisions on known bias variables. During the first of these investigations, we find that the automatic classifiers do not use the color feature as it is specified by the dermoscopic ABCD rule. Hence, third, we investigate different methods to score the color feature. Fourth, we compare the influence of different hyperparameters on the selection of features by the classifiers.

Towards the first question, we find that the different groups of classifiers we investigate in this work, use different subgroups of the features named in the dermoscopic ABCD rule. However, no group of classifiers uses none of the features. While this inspires some trust in the classifiers, we expect an ideal classifier to use all of the features named in this rule. Further, this opens a way to improve the trust in automatic skin lesion classification by enforcing the use of these features. The fact that the asymmetry and border features are used more consistently than

the color and dermoscopic structure features. This indicates a possible inductive bias that the automatic classifier prefers shape over color features. In particular, the color feature, as it is described in the dermoscopic ABCD rule, is used by only one architecture of the melanoma classifiers trained following Perez et al. (2018). This is especially surprising since the color feature is considered one of the most important features by practitioners. However, a negative test only indicates that the deep neural network does not rely on the feature and does not mean that the feature is not useful for the classification task.

Additionally, we found a large difference between the test results using the nonlinear kernel-based Hilbert-Schmidt conditional independence criterion and the linear partial correlation test. The reason for this is that the former indicates nonlinear relationships as, for example, an increase in variance, while the latter is affirmative only if the feature is an indicator for the specific class. This difference is especially visible for the dermoscopic structures feature. This means, that dermoscopic structures most likely make the classification decision more difficult but are not used by the classifier as an indicator of any class.

Towards the second question, we find that all four investigated bias variables, namely the age and sex of a patient as well as the skin color and the existence of colorful patches, are used by at least one group of classifiers. These observations show that more work is needed to create automatic skin lesion classifiers that can be used in practice. One possibility to construct classifiers without including biases that are present in the dataset is the adversarial debiasing method that was introduced in Reimers et al. (2021a) and which we explain in Section 4.2.2.

To investigate the fact, that the state-of-the-art classifiers do not use the color feature, we investigate three additional implementations of the color feature. From these additional implementations, we conclude that not only the hue but also the value and saturation of the pixels within the skin lesion are important.

A common approach in state-of-the-art automatic skin lesion classifiers is the use of large ensemble classifiers. The success of these ensemble classifiers relies on the idea that the individual ensemble members base their decisions on different features of the input image. To this end, we investigate the influence of different hyperparameters on which feature is picked up by the classifier. Our study hints at the architecture being the most important hyperparameter. However, the number of classifiers trained for this study was not large enough to find a significant effect.

In summary, the method described in Section 4.1 is suitable and returns reasonable results for the challenging and relevant real-world application of automatic skin lesion classification, where many important features can not be analyzed using, for example, saliency maps. We find, that all groups of classifiers use at least some of the features named in the dermoscopic ABCD rule, however, all of them additionally rely on at least one bias variable. Finally, we demonstrated how we can apply the method to identify unknown biases and how it can be used to create stronger ensemble classifiers.

5.4 Debiasing

In the previous sections, we summarized the conclusions about the theoretical considerations and the application of the relevant and challenging real-world application of automatic skin lesion classification. As a second application, in Section 4.2.2, we used the new criterion in the context of adversarial debiasing to stop an automatic classifier from picking up a bias that is present in the dataset it is trained on.

Our first conclusion comes from the empirical results in Section 4.3.3 that demonstrate that our new adversarial debiasing method outperforms other adversarial debiasing methods from the literature. This increase in the debiasing performance corroborates our theoretical claims in Section 4.1 and demonstrates that the debiasing losses based on our method can correctly identify whether a feature is used by a deep neural network.

We provided an exact model for the data creation method. This exact model allows us to not only rely on empirical evaluations but also on theoretical arguments. To this end, we mathematically prove that the optimal classifier fulfills our new criterion to determine, whether the classifier ignores the bias feature in its decision, while it does not fulfill the criteria of other methods from the literature.

The experimental results presented in Section 4.3.3 support these theoretical findings. If the reason for a bias in a dataset is correctly modeled by the bias model in Section 4.2.2, then our method explained in Section 4.2.2 is the better choice compared to unconditional debiasing methods presented in the literature. On a toy dataset, that we designed to maximize the difference between the two criteria our method performed significantly better than methods from the literature. Further, the difference in criteria is the reason for the increase in accuracy, and that we showed that this increase extends to classification tasks on real real-world data.

A common drawback of many adversarial debiasing methods is that they decrease the accuracy if they are used erroneously on an unbiased dataset. We are interested in whether the adversarial debiasing method that we described in Section 4.2.2 suffers from the same drawback. For this evaluation, we consider the three subsets of the cats vs dogs example, that are not suitable for adversarial debiasing, namely the unbiased subset (50% in Table 4.21) and the two sets where all dogs are either light- or dark-furred (0% and 100% in Table 4.21). In all three of these experiments, our method performs on par with the baseline and we do not observe a significant drop in performance.

Even though our method outperformed the methods from the literature, it is not able to fully debias the resulting classifier, but only decreases the influence of the bias variable. Multiple possible reasons could be responsible for these observations. First, the independence criterion in our approach is used as a loss and not as a constraint in the optimization. This allows the algorithm to find a solution that is only close to the constraint. Second, since we use an iterative optimization, that only converges to a local minimum and might, hence, not find the optimal, unbiased solution. Third, in contrast to the classification loss, which can be evaluated on a

sample level, the debiasing loss is a property on the level of distributions. Hence, the finite sample effect when evaluating the loss on mini-batches will be very strong for the debiasing loss, which might hinder convergence.

6 | Future Work

In this section, we lay out some further ideas and areas in which this work can be improved. To this end, we start by outlining some ideas to improve the theoretical approach which we introduced in Section 4.1. Then we name some possible future applications where a similar procedure as described in Section 4.2.1 can be applied.

6.1 Identifying Whether a Feature is Relevant for a Supervised Learning Classifier

In this section, we introduce interesting future research directions that can improve the main method described in Section 4.1.

One limitation of our method is that it can not differentiate between features that share the same information. This is a drawback, since semantically distinct features often share some information. For example, in the automatic skin lesion classification task the area of the skin lesion and the length of its border are two features that are semantically distinct but share information because the border of a larger skin lesion will, inevitably, be longer than the border of a small skin lesion.

While this difficulty naturally arises from the use of independence tests, in Section 4.2.1 we already hinted at a possible way to deal with it by using features that are explicitly independent of other features. As an example, when investigating the features named in the dermoscopic ABCD rule, instead of the length of the border we considered the isoperimetric fraction to disrupt the influence of the area of the skin lesion.

However, there is no universal way to separate the information from different semantic features. Hence, an interesting direction of future research is to find a way to distinguish between features that are directly used by a classifier and features whose main influence is due to containing the same information as another feature that is used by the classifier.

As an example related to the study presented in Section 4.2.1, we would consider the length of the border, the area of the skin lesion, the length of the border normalized for the area of the skin lesion, and the area of the skin lesion normalized for the length of the border. The relationship between these four features might reveal insights into the relative importance of the two related features. To evaluate the relationship between the four features, we could rely on an interpretable test

statistic, for example, the conditional mutual information which we introduced in Section 2.3.3.

6.2 Direct Applications of the Method on Deep Neural Network Classifiers

In this Section, we highlight two interesting possibilities for further application of the method presented in Section 4.1 to determine whether specific features are used by a deep learning classifier similar to the study presented in Section 4.2.1. More specifically, the first possibility is to monitor features during the training process and the second is to apply it to a problem from climate science.

6.2.1 Monitoring the Use of Features During the Training Process

The method described in Section 4.1 allows us to understand which feature is used by a deep neural network. An interesting question that can be tackled with this method is to understand at which point during the training process of a deep neural network, it learns to extract a feature. This investigation is motivated by our observation in Section 4.3.3.1, where we found that the number of training epochs influenced the accuracy of the baseline classifier when trained on a biased training set and evaluated on a biased test set. This indicates that at the beginning of the training, a different feature is used than at the end of the training process.

To this end, it would be interesting to train multiple, different classifiers for a problem, for example, the problem presented in Section 4.2.1, and carry out the analysis we presented in Section 4.3.2 after every epoch.

Further, not only the effect of the number of training epochs but the effect of all hyperparameters on the selection of extracted features should be investigated. To this end, interesting questions include, whether hyperparameters can lead to neural networks preferring local patterns over global properties of images, whether hyperparameters can lead to a more sparse feature representation, or whether hyperparameters make the neural network prefer features of higher complexity over simpler features.

6.2.2 Applying Our Method to Further Situations

The application of automatic skin lesion classification leads to interesting insights into the advantages and drawbacks of the method presented in Section 4.1 and also into state-of-the-art deep neural network classifiers performing skin lesion classification. The success of this study relies on the fact, that we have prior knowledge on how to solve the problem, in particular, which features are relevant to reach a prediction. In the example, in Section 4.2.1 this prior knowledge is given by the dermoscopic ABCD rule. To this end, it would be interesting to apply the

method to other tasks where such prior knowledge exists. Some tasks that fulfill this requirement are situated in climate science. For example, the authors of Kretschmer et al. (2017), use methods from causal inference to identify meaningful features, more specifically, averages over some regions where a variable behaves consistently. The authors of that paper identify 471 possible regions of which they, in the end, consider only four to be relevant. It would be very interesting to determine, whether a deep neural network predictor trained on the same data would select the same or similar features.

6.3 Adversarial Debiasing

In this section, we lay out possible ways to improve the adversarial debiasing method described in Section 4.2.2. The first idea is to apply the analysis to other bias models than the one presented in Section 4.2.2.2. We describe this idea further in Section 6.3.1. The second idea is to extend the proof in Section 4.2.2. We described this idea further in Section 6.3.2. Third, we describe further experiments that could increase the trust in our method in Section 6.3.3. Finally, we outline one methodical direction of future work in Section 6.3.4.

6.3.1 Applying the Debiasing Method to a Wider Range of Biases

In Section 4.2.2, we described the data creation method that leads to the bias we consider in this work. Even though this bias model covers many known biases in computer vision, it does not cover all of them. One straightforward way to extend the work presented here is to model the data creation methods for other kinds of biases as used, for example, in algorithmic fairness, where standard debiasing methods often are not the best solution, as discussed, for example, in Liu et al. (2018a).

6.3.2 Extending the Proof in Section 4.2.2

In Section 4.2.2, we stated that if the data is related as displayed in the structural causal model in Figure 4.11 then the optimal classifier fulfills our proposed conditional independence criterion

$$P \perp\!\!\!\perp B \,|\, L \tag{6.1}$$

but does not fulfill the unconditional independence criterion

$$P \perp\!\!\!\perp B \tag{6.2}$$

used throughout the related work in the literature. However, in that section, we present mathematical proof only for a simple linear case. The method we used to

proof that result should be extendable to the general case. Extending this proof is an important future research direction.

6.3.3 Additional Experiments for Adversarial Debiasing

The experiments we presented in Section 4.3.3 corroborate the claims made in Section 4.2.2. However, more experiments could be useful to further explore the use of our method and to ensure, that the method is applicable to a wide range of computer vision tasks. Apart from the extremely simple regression experiment presented in Section 4.3.3.1, all experiments described in Section 4.3.3 are binary classification tasks. To this end, more experiments on tasks that are either multiclass classification or real-world regression problems would either give us additional insurance or highlight possible shortcomings of our method. Further, in the two classification tasks, the box vs cross classification in Section 4.3.3.1 and the cats vs dogs experiment in Section 4.3.3.3 the difference between the bias signal and the meaningful system is that one is a color and the other is a structure feature. To make sure, that our method is also applicable if the difference between these Signals is different we want to conduct further experiments where both signals are structure or color signals. Finally, while the cats vs dogs experiment is carried out on real-world images of cats and dogs, hand-labeled as either light-furred or dark-furred, the problem is scientific and has no relevance in practice. Consequently, one could argue, that it does not have the same complexity as relevant real-world problems. To this end, another direction of future work is to apply the method to a relevant real-world problem, for example, the problem of automatic skin lesion classification introduced in Section 4.2.1.

6.3.4 Handling the Difficulties of Estimating the Loss from Minibatches

As mentioned in Section 5.4 the debiasing loss is, in contrast to the classification loss, not calculated for each sample but instead for the whole dataset. This results in a loss that has a different magnitude and variance across mini-batches. While the former can be solved by a scaling factor, the latter is more difficult to handle. As a direction of future research, we would like to understand the effect of this variance, as well as ways to mitigate this effect.

Bibliography

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org. (cited on page 130)

Adebayo, J., Gilmer, J., Goodfellow, I., and Kim, B. (2018). Local explanation methods for deep neural networks lack sensitivity to parameter values. *arXiv preprint arXiv:1810.03307*. (cited on pages 46, 100, and 106)

Adeli, E., Zhao, Q., Pfefferbaum, A., Sullivan, E. V., Fei-Fei, L., Niebles, J. C., and Pohl, K. M. (2021). Representation learning with statistical independence to mitigate bias. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2513–2523. (cited on pages 42, 119, 128, 144, 146, 147, 148, 149, and 150)

Agarwal, C. and Nguyen, A. (2020). Explaining image classifiers by removing input features using generative models. In *Proceedings of the Asian Conference on Computer Vision*. (cited on page 45)

Ali Shah, S. A., Uddin, I., Aziz, F., Ahmad, S., Al-Khasawneh, M. A., and Sharaf, M. (2020). An enhanced deep neural network for predicting workplace absenteeism. *Complexity*, 2020. (cited on page 122)

Alman, J. and Williams, V. V. (2021). A refined laser method and faster matrix multiplication. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 522–539. SIAM. (cited on page 40)

Altman, N. S. (1992). An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185. (cited on page 4)

Alvi, M., Zisserman, A., and Nellåker, C. (2018). Turning a blind eye: Explicit removal of biases and variation from deep neural network embeddings. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0. (cited on pages 119 and 125)

Andriushchenko, M., Croce, F., Flammarion, N., and Hein, M. (2020). Square attack: a query-efficient black-box adversarial attack via random search. In *European Conference on Computer Vision*, pages 484–501. Springer. (cited on page 53)

Angwin, J., Larson, J., Mattu, S., and Kirchner, L. (2016). Machine bias – there's software used across the country to predict future criminals. and it's biased against blacks. *Pro Publica*. (cited on page 13)

Argenziano, G., Fabbrocini, G., Carli, P., De Giorgi, V., Sammarco, E., and Delfino, M. (1998). Epiluminescence microscopy for the diagnosis of doubtful melanocytic skin lesions: comparison of the abcd rule of dermatoscopy and a new 7-point checklist based on pattern analysis. *Archives of dermatology*, 134(12):1563–1570. (cited on page 113)

Baba, K., Shibata, R., and Sibuya, M. (2004). Partial correlation and conditional correlation as measures of conditional independence. *Australian & New Zealand Journal of Statistics*, 46(4):657–664. (cited on page 35)

Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.-R., and Samek, W. (2015). On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140. (cited on page 46)

Bahmer, F. A., Fritsch, P., Kreusch, J., Pehamberger, H., Rohrer, C., Schindera, I., Smolle, J., Soyer, H. P., and Stolz, W. (1990). Terminology in surface microscopy: consensus meeting of the committee on analytical morphology of the arbeitsgemeinschaft dermatologische forschung, hamburg, federal republic of germany, nov. 17, 1989. *Journal of the American Academy of Dermatology*, 23(6):1159–1162. (cited on page 1)

Bai, W., Quan, C., and Luo, Z. (2017). Alleviating adversarial attacks via convolutional autoencoder. In *2017 18th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, pages 53–58. IEEE. (cited on pages 9, 10, and 55)

Barber, R. F. and Candès, E. J. (2015). Controlling the false discovery rate via knockoffs. *The Annals of Statistics*, 43(5):2055–2085. (cited on page 45)

Barry-Jester, A. M., Casselman, B., and Goldstein, D. (2015). The new science of sentencing. *The Marshall Project*, 4:2015. (cited on page 2)

Bell, J. S. (1964). On the einstein podolsky rosen paradox. *Physics Physique Fizika*, 1(3):195. (cited on pages 25 and 27)

Berkeley, G. (1881). *A treatise concerning the principles of human knowledge*. JB Lippincott & Company. (cited on pages 25 and 26)

Bissoto, A., Fornaciali, M., Valle, E., and Avila, S. (2019). (De) Constructing Bias on Skin Lesion Datasets. In *2019 IEEE/CVF Conference on Computer Vision and*

Pattern Recognition Workshops (CVPRW), pages 2766–2774, Long Beach, CA, USA. IEEE. (cited on pages 12 and 108)

Bissoto, A., Valle, E., and Avila, S. (2020). Debiasing skin lesion datasets and models? not so fast. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 740–741. (cited on pages 6 and 108)

Brendel, W., Rauber, J., and Bethge, M. (2017). Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. *arXiv preprint arXiv:1712.04248*. (cited on page 53)

Brinker, T. J., Hekler, A., Enk, A. H., Klode, J., Hauschild, A., Berking, C., Schilling, B., Haferkamp, S., Schadendorf, D., Holland-Letz, T., Utikal, J. S., von Kalle, C., Ludwig-Peitsch, W., Sirokay, J., Heinzerling, L., Albrecht, M., Baratella, K., Bischof, L., Chorti, E., Dith, A., Drusio, C., Giese, N., Gratsias, E., Griewank, K., Hallasch, S., Hanhart, Z., Herz, S., Hohaus, K., Jansen, P., Jockenhöfer, F., Kanaki, T., Knispel, S., Leonhard, K., Martaki, A., Matei, L., Matull, J., Olischewski, A., Petri, M., Placke, J.-M., Raub, S., Salva, K., Schlott, S., Sody, E., Steingrube, N., Stoffels, I., Ugurel, S., Zaremba, A., Gebhardt, C., Booken, N., Christolouka, M., Buder-Bakhaya, K., Bokor-Billmann, T., Enk, A., Gholam, P., Hänßle, H., Salzmann, M., Schäfer, S., Schäkel, K., Schank, T., Bohne, A.-S., Deffaa, S., Drerup, K., Egberts, F., Erkens, A.-S., Ewald, B., Falkvoll, S., Gerdes, S., Harde, V., Hauschild, A., Jost, M., Kosova, K., Messinger, L., Metzner, M., Morrison, K., Motamedi, R., Pinczker, A., Rosenthal, A., Scheller, N., Schwarz, T., Stölzl, D., Thielking, F., Tomaschewski, E., Wehkamp, U., Weichenthal, M., Wiedow, O., Bär, C. M., Bender-Säbelkampf, S., Horbrügger, M., Karoglan, A., Kraas, L., Faulhaber, J., Geraud, C., Guo, Z., Koch, P., Linke, M., Maurier, N., Müller, V., Thomas, B., Utikal, J. S., Alamri, A. S. M., Baczako, A., Berking, C., Betke, M., Haas, C., Hartmann, D., Heppt, M. V., Kilian, K., Krammer, S., Lapczynski, N. L., Mastnik, S., Nasifoglu, S., Ruini, C., Sattler, E., Schlaak, M., Wolff, H., Achatz, B., Bergbreiter, A., Drexler, K., Ettinger, M., Haferkamp, S., Halupczok, A., Hegemann, M., Dinauer, V., Maagk, M., Mickler, M., Philipp, B., Wilm, A., Wittmann, C., Gesierich, A., Glutsch, V., Kahlert, K., Kerstan, A., Schilling, B., and Schrüfer, P. (2019). Deep learning outperformed 136 of 157 dermatologists in a head-to-head dermoscopic melanoma image classification task. *European Journal of Cancer*, 113:47–54. (cited on pages 11 and 12)

Carlini, N. and Wagner, D. (2017). Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. IEEE. (cited on pages 9, 52, 53, and 67)

Carlini, N. and Wagner, D. (2018). Audio adversarial examples: Targeted attacks on speech-to-text. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 1–7. IEEE. (cited on page 54)

Caruana, R., Kangarloo, H., Dionisio, J. D., Sinha, U., and Johnson, D. (1999). Case-based explanation of non-case-based learning methods. In *Proceedings of the*

AMIA Symposium, page 212. American Medical Informatics Association. (cited on page 49)

Celebi, M. E., Kingravi, H. A., Uddin, B., Iyatomi, H., Aslandogan, Y. A., Stoecker, W. V., and Moss, R. H. (2007). A methodological approach to the classification of dermoscopy images. *Computerized Medical imaging and graphics*, 31(6):362–373. (cited on pages 1, 11, and 114)

Chalupka, K., Perona, P., and Eberhardt, F. (2018). Fast conditional independence test for vector variables with large sample sizes. *arXiv preprint arXiv:1804.02747*. (cited on pages 36 and 130)

Chen, J., Jordan, M. I., and Wainwright, M. J. (2020). Hopskipjumpattack: A query-efficient decision-based attack. In *2020 ieee symposium on security and privacy (sp)*, pages 1277–1294. IEEE. (cited on page 53)

Chen, P.-Y., Sharma, Y., Zhang, H., Yi, J., and Hsieh, C.-J. (2018). Ead: elastic-net attacks to deep neural networks via adversarial examples. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32. (cited on page 53)

Chen, P.-Y., Zhang, H., Sharma, Y., Yi, J., and Hsieh, C.-J. (2017). Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM workshop on artificial intelligence and security*, pages 15–26. (cited on page 53)

Chen, Z.-M., Wei, X.-S., Wang, P., and Guo, Y. (2019). Multi-label image recognition with graph convolutional networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5177–5186. (cited on pages 132 and 133)

Codella, N., Rotemberg, V., Tschandl, P., Celebi, M. E., Dusza, S., Gutman, D., Helba, B., Kalloo, A., Liopyris, K., Marchetti, M., Kittler, H., and Halpern, A. (2019). Skin Lesion Analysis Toward Melanoma Detection 2018: A Challenge Hosted by the International Skin Imaging Collaboration (ISIC). *arXiv:1902.03368 [cs]*. arXiv: 1902.03368. (cited on pages 2, 5, and 139)

Codella, N. C., Gutman, D., Celebi, M. E., Helba, B., Marchetti, M. A., Dusza, S. W., Kalloo, A., Liopyris, K., Mishra, N., Kittler, H., et al. (2018). Skin lesion analysis toward melanoma detection: A challenge at the 2017 international symposium on biomedical imaging (isbi), hosted by the international skin imaging collaboration (isic). In *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, pages 168–172. IEEE. (cited on pages 1, 6, 109, 110, 113, 116, 141, 186, and 188)

Combalia, M., Codella, N. C. F., Rotemberg, V., Helba, B., Vilaplana, V., Reiter, O., Carrera, C., Barreiro, A., Halpern, A. C., Puig, S., and Malvehy, J. (2019). BCN20000: Dermoscopic Lesions in the Wild. *arXiv:1908.02288 [cs, eess]*. arXiv: 1908.02288. (cited on pages 1 and 110)

Cubuk, E. D., Zoph, B., Schoenholz, S. S., and Le, Q. V. (2017). Intriguing properties of adversarial examples. *arXiv preprint arXiv:1711.02846*. (cited on page 9)

Cui, Y., Song, Y., Sun, C., Howard, A., and Belongie, S. (2018). Large scale fine-grained categorization and domain-specific transfer learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4109–4118. (cited on page 134)

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314. (cited on pages 32 and 33)

Dabkowski, P. and Gal, Y. (2017). Real time image saliency for black box classifiers. *Advances in neural information processing systems*, 30. (cited on page 45)

Dalkin, S. M., Greenhalgh, J., Jones, D., Cunningham, B., and Lhussier, M. (2015). What's in a mechanism? development of a key concept in realist evaluation. *Implementation science*, 10(1):1–7. (cited on page 26)

Dalvi, N., Domingos, P., Sanghai, S., and Verma, D. (2004). Adversarial classification. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 99–108. (cited on pages 9 and 54)

Dong, Y., Su, H., Zhu, J., and Bao, F. (2017). Towards interpretable deep neural networks by leveraging adversarial examples. *arXiv preprint arXiv:1708.05493*. (cited on pages 9 and 51)

Dwork, C., Hardt, M., Pitassi, T., Reingold, O., and Zemel, R. (2012). Fairness through awareness. In *Proceedings of the 3rd innovations in theoretical computer science conference*, pages 214–226. (cited on page 142)

Ebrahimi, J., Rao, A., Lowd, D., and Dou, D. (2017). Hotflip: White-box adversarial examples for text classification. *arXiv preprint arXiv:1712.06751*. (cited on page 54)

Egert, C. (2012). Lineare statistische modellierung und interpretation in der praxis. In *Lineare statistische Modellierung und Interpretation in der Praxis*. Oldenbourg Wissenschaftsverlag. (cited on page 41)

Engstrom, L., Tran, B., Tsipras, D., Schmidt, L., and Madry, A. (2019). Exploring the landscape of spatial robustness. In *International Conference on Machine Learning*, pages 1802–1811. PMLR. (cited on page 67)

Erhan, D., Bengio, Y., Courville, A., and Vincent, P. (2009). Visualizing higher-layer features of a deep network. *University of Montreal*, 1341(3):1. (cited on pages 9, 48, 50, and 118)

Esteva, A., Kuprel, B., Novoa, R. A., Ko, J., Swetter, S. M., Blau, H. M., and Thrun, S. (2017). Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542(7639):115–118. Number: 7639 Publisher: Nature Publishing Group. (cited on page 11)

Everingham, M., Van Gool, L., Williams, C. K., Winn, J., and Zisserman, A. (2010). The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338. (cited on page 66)

Eykholt, K., Evtimov, I., Fernandes, E., Kohno, T., Li, B., Prakash, A., Rahmati, A., and Song, D. (2017). Robust physical-world attacks on machine learning models. *arXiv preprint arXiv:1707.08945*, 2(3):4. (cited on pages 9 and 56)

Fawzi, A., Fawzi, O., and Frossard, P. (2015). Fundamental limits on adversarial robustness. In *Proc. ICML, Workshop on Deep Learning*. (cited on page 55)

Fawzi, A., Fawzi, O., and Frossard, P. (2018). Analysis of classifiers' robustness to adversarial perturbations. *Machine Learning*, 107(3):481–508. (cited on page 55)

Fawzi, A., Moosavi-Dezfooli, S.-M., and Frossard, P. (2016). Robustness of classifiers: from adversarial to random noise. *arXiv preprint arXiv:1608.08967*. (cited on page 55)

Fisher, R. (1925). Statistical methods for research workers, 13e. *London: Oliver and Loyd, Ltd*, pages 99–101. (cited on page 33)

Fletcher, R. (2013). *Practical methods of optimization*. John Wiley & Sons. (cited on page 53)

Fukumizu, K., Gretton, A., Sun, X., and Schölkopf, B. (2007). Kernel measures of conditional dependence. In *NIPS*, volume 20, pages 489–496. (cited on pages 40 and 119)

Geller, A. C., Swetter, S. M., Brooks, K., Demierre, M.-F., and Yaroch, A. L. (2007). Screening, early detection, and trends for melanoma: current status (2000-2006) and future directions. *Journal of the American Academy of Dermatology*, 57(4):555–572; quiz 573–576. (cited on page 11)

Gerhardus, A. and Runge, J. (2020). High-recall causal discovery for autocorrelated time series with latent confounders. *arXiv preprint arXiv:2007.01884*. (cited on page 25)

Gessert, N., Nielsen, M., Shaikh, M., Werner, R., and Schlaefer, A. (2020). Skin lesion classification using ensembles of multi-resolution EfficientNets with meta data. *MethodsX*, 7:100864. (cited on pages 2, 5, 11, 12, 13, 109, 110, 135, 136, 137, 138, 139, 140, 189, and 190)

Ghiasi, A., Shafahi, A., and Goldstein, T. (2020). Breaking certified defenses: Semantic adversarial examples with spoofed robustness certificates. *arXiv preprint arXiv:2003.08937*. (cited on page 53)

Ghorbani, A., Abid, A., and Zou, J. (2019). Interpretation of neural networks is fragile. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3681–3688. (cited on pages 46, 100, and 106)

Gilmer, J., Adams, R. P., Goodfellow, I., Andersen, D., and Dahl, G. E. (2018a). Motivating the rules of the game for adversarial example research. *arXiv preprint arXiv:1807.06732*. (cited on pages 9 and 57)

Gilmer, J., Ford, N., Carlini, N., and Cubuk, E. (2019). Adversarial examples are a natural consequence of test error in noise. In *International Conference on Machine Learning*, pages 2280–2289. PMLR. (cited on pages 55 and 59)

Gilmer, J., Metz, L., Faghri, F., Schoenholz, S. S., Raghu, M., Wattenberg, M., and Goodfellow, I. (2018b). Adversarial spheres. *arXiv preprint arXiv:1801.02774*. (cited on pages 56 and 59)

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014a). Generative adversarial nets. *Advances in neural information processing systems*, 27. (cited on pages 45 and 47)

Goodfellow, I. J., Shlens, J., and Szegedy, C. (2014b). Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*. (cited on pages 9, 10, 52, 53, 54, 58, 67, and 135)

Goyal, Y., Feder, A., Shalit, U., and Kim, B. (2019). Explaining classifiers with causal concept effect (cace). *arXiv preprint arXiv:1907.07165*. (cited on pages 15, 16, 47, 103, 106, and 107)

Granger, C. W. (1969). Investigating causal relations by econometric models and cross-spectral methods. *Econometrica: journal of the Econometric Society*, pages 424–438. (cited on pages 25, 26, and 27)

Gretton, A., Fukumizu, K., Teo, C. H., Song, L., Schölkopf, B., Smola, A. J., et al. (2007). A kernel statistical test of independence. In *Nips*, volume 20, pages 585–592. Citeseer. (cited on pages 14, 39, 119, and 126)

Grosse, K., Pfaff, D., Smith, M. T., and Backes, M. (2018). The limitations of model uncertainty in adversarial settings. *arXiv preprint arXiv:1812.02606*. (cited on page 53)

Gu, S. and Rigazio, L. (2014). Towards deep neural network architectures robust to adversarial examples. *arXiv preprint arXiv:1412.5068*. (cited on pages 9, 10, 55, and 62)

Guo, C., Gardner, J., You, Y., Wilson, A. G., and Weinberger, K. (2019). Simple black-box adversarial attacks. In *International Conference on Machine Learning*, pages 2484–2493. PMLR. (cited on page 53)

Hall, N. (2004). Two concepts of causation. *Causation and counterfactuals*, pages 225–276. (cited on page 28)

Hardt, M., Price, E., and Srebro, N. (2016). Equality of opportunity in supervised learning. *arXiv preprint arXiv:1610.02413*. (cited on page 142)

Harradon, M., Druce, J., and Ruttenberg, B. (2018). Causal learning and explanation of deep neural networks via autoencoded activations. *arXiv preprint arXiv:1802.00541*. (cited on page 48)

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep Residual Learning for Image Recognition. *arXiv:1512.03385 [cs]*. arXiv: 1512.03385. (cited on page 109)

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778. (cited on pages 51, 150, and 183)

Hill, A. B. (1965). The environment and disease: association or causation? (cited on pages 25, 26, and 27)

Hitchcock, C. (2001). The intransitivity of causation revealed in equations and graphs. *The Journal of Philosophy*, 98(6):273–299. (cited on page 28)

Ho, T. K. (1995). Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pages 278–282. IEEE. (cited on page 4)

Hoffmann, K., Gambichler, T., Rick, A., Kreutz, M., Anschuetz, M., Grünendick, T., Orlikov, A., Gehlen, S., Perotti, R., Andreassi, L., et al. (2003). Diagnostic and neural analysis of skin cancer (danaos). a multicentre study for collection and computer-aided analysis of data from pigmented skin lesions using digital dermoscopy. *British Journal of Dermatology*, 149(4):801–809. (cited on page 1)

Hornik, K. (1991). Approximation capabilities of multilayer feedforward network. *Neural networks*, 4(2):251–257. (cited on page 33)

Hosseini, H. and Poovendran, R. (2018). Semantic adversarial examples. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1614–1619. (cited on page 67)

Huang, G., Liu, Z., van der Maaten, L., and Weinberger, K. Q. (2018). Densely Connected Convolutional Networks. *arXiv:1608.06993 [cs]*. arXiv: 1608.06993. (cited on page 109)

Hume, D. (1896). *A treatise of human nature*. Clarendon Press. (cited on pages 25 and 26)

ISIC-archive (2021). International skin imaging collaboration, ISIC Archive. `https://www.isic-archive.com/`. (cited on page 111)

Jang, U., Wu, X., and Jha, S. (2017). Objective metrics and gradient descent algorithms for adversarial examples in machine learning. In *Proceedings of the 33rd Annual Computer Security Applications Conference*, pages 262–277. (cited on page 53)

Kallenberg, O. (1997). *Foundations of modern probability*, volume 2. Springer. (cited on page 36)

Kannan, H., Kurakin, A., and Goodfellow, I. (2018). Adversarial logit pairing. *arXiv preprint arXiv:1803.06373*. (cited on pages 9 and 10)

Kim, B., Kim, H., Kim, K., Kim, S., and Kim, J. (2019). Learning not to learn: Training deep neural networks with biased data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9012–9020. (cited on pages 118, 119, 126, 144, 146, 147, 149, 150, and 190)

Kim, B., Wattenberg, M., Gilmer, J., Cai, C., Wexler, J., Viegas, F., et al. (2018). Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *International conference on machine learning*, pages 2668–2677. PMLR. (cited on pages 15, 16, and 46)

Kindermans, P.-J., Hooker, S., Adebayo, J., Alber, M., Schütt, K. T., Dähne, S., Erhan, D., and Kim, B. (2019). The (un) reliability of saliency methods. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, pages 267–280. Springer. (cited on pages 46, 100, and 106)

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*. (cited on pages 145 and 150)

Kretschmer, M., Runge, J., and Coumou, D. (2017). Early prediction of extreme stratospheric polar vortex states based on causal precursors. *Geophysical research letters*, 44(16):8592–8600. (cited on page 163)

Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images. (cited on pages 62, 63, 70, 71, and 183)

Kurakin, A., Goodfellow, I., Bengio, S., et al. (2016). Adversarial examples in the physical world. (cited on pages 53 and 62)

Lakkaraju, H., Kamar, E., Caruana, R., and Horvitz, E. (2016). Discovering blind spots of predictive models: Representations and policies for guided exploration. *arXiv preprint arXiv*, 1610. (cited on pages 142 and 149)

Lapuschkin, S., Wäldchen, S., Binder, A., Montavon, G., Samek, W., and Müller, K.-R. (2019). Unmasking clever hans predictors and assessing what machines really learn. *Nature communications*, 10(1):1–8. (cited on pages 6, 13, 46, 66, 106, and 120)

Larson, J., Mattu, S., Kirchner, L., and Angwin, J. (2016). How we analyzed the compas recidivism algorithm. (cited on page 2)

LeCun, Y. (1998). The mnist database of handwritten digits. *http://yann.lecun.com/exdb/mnist/*. (cited on pages 56, 62, 70, 71, and 73)

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324. (cited on page 111)

Leshno, M., Lin, V. Y., Pinkus, A., and Schocken, S. (1993). Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural networks*, 6(6):861–867. (cited on page 33)

Li, X. and Li, F. (2017). Adversarial examples detection in deep networks with convolutional filter statistics. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5764–5772. (cited on pages 9, 10, and 55)

Li, Y., Bai, S., Xie, C., Liao, Z., Shen, X., and Yuille, A. L. (2019a). Regional homogeneity: Towards learning transferable universal adversarial perturbations against defenses. *arXiv preprint arXiv:1904.00979*. (cited on page 56)

Li, Z., Perez-Suay, A., Camps-Valls, G., and Sejdinovic, D. (2019b). Kernel dependence regularizers and gaussian processes with applications to algorithmic fairness. *arXiv preprint arXiv:1911.04322*. (cited on pages 119 and 127)

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer. (cited on pages 5, 129, 132, and 155)

Liu, L. T., Dean, S., Rolf, E., Simchowitz, M., and Hardt, M. (2018a). Delayed impact of fair machine learning. In *International Conference on Machine Learning*, pages 3150–3158. PMLR. (cited on page 163)

Liu, X., Yang, H., Liu, Z., Song, L., Li, H., and Chen, Y. (2018b). Dpatch: An adversarial patch attack on object detectors. *arXiv preprint arXiv:1806.02299*. (cited on page 53)

Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440. (cited on page 13)

Loshchilov, I. and Hutter, F. (2016). Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*. (cited on page 150)

Lowd, D. and Meek, C. (2005). Adversarial learning. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 641–647. (cited on pages 9 and 54)

Lu, J., Issaranon, T., and Forsyth, D. (2017). Safetynet: Detecting and rejecting adversarial examples robustly. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 446–454. (cited on pages 9, 10, and 55)

Luo, Y., Boix, X., Roig, G., Poggio, T., and Zhao, Q. (2015). Foveation-based mechanisms alleviate adversarial examples. *arXiv preprint arXiv:1511.06292*. (cited on page 67)

Mackie, J. L. (1965). Causes and conditions. *American philosophical quarterly*, 2(4):245–264. (cited on pages 25, 26, and 27)

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. (2017). Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*. (cited on pages 9, 10, 51, 52, 53, 62, and 183)

Metzen, J. H., Chaithanya Kumar, M., Brox, T., and Fischer, V. (2017). Universal adversarial perturbations against semantic image segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2755–2764. (cited on pages 9, 56, 57, and 59)

Mishra, N. K. and Celebi, M. E. (2016). An Overview of Melanoma Detection in Dermoscopy Images Using Image Processing and Machine Learning. *arXiv:1601.07843 [cs, stat]*. arXiv: 1601.07843. (cited on pages 12, 13, and 108)

Montavon, G., Lapuschkin, S., Binder, A., Samek, W., and Müller, K.-R. (2017). Explaining nonlinear classification decisions with deep taylor decomposition. *Pattern Recognition*, 65:211–222. (cited on page 46)

Moosavi-Dezfooli, S.-M., Fawzi, A., Fawzi, O., and Frossard, P. (2017). Universal adversarial perturbations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1765–1773. (cited on pages 9, 56, and 59)

Moosavi-Dezfooli, S.-M., Fawzi, A., Fawzi, O., Frossard, P., and Soatto, S. (2019). Analysis of universal adversarial perturbations. *arXiv preprint arXiv:1705.09554*. (cited on pages 56 and 58)

Moosavi-Dezfooli, S.-M., Fawzi, A., and Frossard, P. (2016). Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2574–2582. (cited on pages 9, 11, 52, 53, 71, and 73)

Mopuri, K. R., Garg, U., and Babu, R. V. (2018). Cnn fixations: an unraveling approach to visualize the discriminative image regions. *IEEE Transactions on Image Processing*, 28(5):2116–2125. (cited on page 46)

Mordvintsev, A., Olah, C., and Tyka, M. (2015). Inceptionism: Going deeper into neural networks. (cited on pages 48 and 49)

Muckatira, S. (2020). Properties Of Winning Tickets On Skin Lesion Classification. *arXiv:2008.12141 [cs, eess]*. arXiv: 2008.12141. (cited on pages 6, 12, 13, and 108)

Nachbar, F., Stolz, W., Merkle, T., Cognetta, A. B., Vogt, T., Landthaler, M., Bilek, P., Braun-Falco, O., and Plewig, G. (1994). The ABCD rule of dermatoscopy. High prospective value in the diagnosis of doubtful melanocytic skin lesions. *Journal of the American Academy of Dermatology*, 30(4):551–559. (cited on pages 1, 5, 12, 105, 111, 112, and 114)

Narendra, T., Sankaran, A., Vijaykeerthy, D., and Mani, S. (2018). Explaining deep learning models using causal inference. *arXiv preprint arXiv:1811.04376*. (cited on page 48)

Niemann, H. (1990). *Pattern analysis and understanding*, volume 2. Springer Science & Business Media. (cited on pages 51, 52, and 82)

Olah, C., Mordvintsev, A., and Schubert, L. (2017). Feature visualization. *Distill*. https://distill.pub/2017/feature-visualization. (cited on pages 48, 49, and 50)

Papernot, N., McDaniel, P., and Goodfellow, I. (2016a). Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*. (cited on pages 53 and 56)

Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z. B., and Swami, A. (2016b). The limitations of deep learning in adversarial settings. In *2016 IEEE European symposium on security and privacy (EuroS&P)*, pages 372–387. IEEE. (cited on page 53)

Papernot, N., McDaniel, P., Wu, X., Jha, S., and Swami, A. (2016c). Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE symposium on security and privacy (SP)*, pages 582–597. IEEE. (cited on pages 9 and 10)

Pearl, J. (2009). *Causality*. Cambridge university press. (cited on pages I, 4, 14, 19, 21, 22, 24, 25, 27, 28, 29, 77, 92, and 106)

Perez, F., Vasconcelos, C., Avila, S., and Valle, E. (2018). Data augmentation for skin lesion analysis. In *OR 2.0 Context-Aware Operating Theaters, Computer Assisted Robotic Endoscopy, Clinical Image-Based Procedures, and Skin Image Analysis*, pages 303–311. Springer. (cited on pages 2, 5, 11, 12, 13, 108, 109, 110, 135, 136, 137, 138, 139, 140, 141, 142, 156, 157, 188, 189, and 190)

Pérez-Suay, A., Laparra, V., Mateo-García, G., Muñoz-Marí, J., Gómez-Chova, L., and Camps-Valls, G. (2017). Fair kernel learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 339–355. Springer. (cited on pages 119 and 127)

Peters, J., Janzing, D., and Schölkopf, B. (2017). *Elements of causal inference: foundations and learning algorithms*. The MIT Press. (cited on pages 4, 14, 19, 22, and 24)

Pinkus, A. (1999). Approximation theory of the mlp model in neural networks. *Acta numerica*, 8:143–195. (cited on page 33)

Polyak, B. T. (1964). Some methods of speeding up the convergence of iteration methods. *Ussr computational mathematics and mathematical physics*, 4(5):1–17. (cited on page 111)

Popescu, O.-I., Shadaydeh, M., and Denzler, J. (2021). Counterfactual generation with knockoffs. *arXiv preprint arXiv:2102.00951*. (cited on page 45)

Reichenbach, H. (1991). *The direction of time*, volume 65. Univ of California Press. (cited on pages 4, 24, 25, 27, 77, and 106)

Reimers, C., Bodesheim, P., Runge, J., and Denzler, J. (2021a). Towards Learning an Unbiased Classifier from Biased Data via Conditional Adversarial Debiasing. *arXiv:2103.06179 [cs]*. arXiv: 2103.06179. (cited on pages 8, 116, 117, 120, 122, 142, 144, 147, 148, 149, 157, 186, and 190)

Reimers, C., Penzel, N., Bodesheim, P., Runge, J., and Denzler, J. (2021b). Conditional dependence tests reveal the usage of abcd rule features and bias variables in automatic skin lesion classification. accepted for publication, added to the sublementary for review. (cited on pages 6, 106, 109, 110, 136, 137, 138, 139, 188, 189, and 190)

Reimers, C. and Requena-Mesa, C. (2020). Deep learning–an opportunity and a challenge for geo-and astrophysics. In *Knowledge Discovery in Big Data from Astronomy and Earth Observation*, pages 251–265. (cited on pages 12, 32, 43, and 117)

Reimers, C., Runge, J., and Denzler, J. (2019). Using causal inference to globally understand black box predictors beyond saliency maps. In *International Workshop on Climate Informatics (CI)*. (cited on pages 4, 99, 129, 131, and 188)

Reimers, C., Runge, J., and Denzler, J. (2020). Determining the relevance of features for deep neural networks. In *European Conference on Computer Vision*, pages 330–346. Springer. (cited on pages 5, 89, 90, 98, 132, 133, 134, 185, 186, and 188)

Rieger, L., Singh, C., Murdoch, W. J., and Yu, B. (2020). Interpretations are useful: penalizing explanations to align neural networks with prior knowledge. *arXiv:1909.13584 [cs, stat]*. arXiv: 1909.13584. (cited on pages 12, 13, 108, and 116)

Rozsa, A., Günther, M., and Boult, T. E. (2016a). Are accuracy and robustness correlated. In *2016 15th IEEE international conference on machine learning and applications (ICMLA)*, pages 227–232. IEEE. (cited on pages 10, 55, 56, and 59)

Rozsa, A., Gunther, M., and Boult, T. E. (2018). Towards robust deep neural networks with bang. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 803–811. IEEE. (cited on pages 9 and 10)

Rozsa, A., Rudd, E. M., and Boult, T. E. (2016b). Adversarial diversity and hard positive generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 25–32. (cited on pages 9, 10, 58, and 67)

Runge, J. (2020). Discovering contemporaneous and lagged causal relations in autocorrelated nonlinear time series datasets. *arXiv preprint arXiv:2003.03685*. (cited on page 25)

Runge, J., Nowack, P., Kretschmer, M., Flaxman, S., and Sejdinovic, D. (2019). Detecting and quantifying causal associations in large nonlinear time series datasets. *Science Advances*, 5(11):eaau4996. (cited on page 25)

Russakovsky, O., Deng, J., Huang, Z., Berg, A. C., and Fei-Fei, L. (2013). Detecting avocados to zucchinis: what have we done, and where are we going? In *Proceedings*

of the IEEE International Conference on Computer Vision, pages 2064–2071. (cited on pages 12 and 132)

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252. (cited on pages 2, 13, 51, 54, 55, 67, 109, 110, 151, and 183)

Russell, B. (2013). *Mysticism and logic*. Courier Corporation. (cited on page 26)

San Liang, X. (2014). Unraveling the cause-effect relation between time series. *Physical Review E*, 90(5):052150. (cited on page 27)

Sarmanov, O. V. (1958). The maximum correlation coefficient (symmetrical case). In *Doklady Akademii Nauk*, volume 120, pages 715–718. Russian Academy of Sciences. (cited on pages 14 and 36)

Schneider, F., Hartmann, B., Raste, T., Kretschmann, M., Amthor, M., and Denzler, J. (2018). Aquaplaning - a potential hazard also for automated driving. (cited on pages 2, 7, and 121)

Scope, A., Marchetti, M. A., Marghoob, A. A., Dusza, S. W., Geller, A. C., Satagopan, J. M., Weinstock, M. A., Berwick, M., and Halpern, A. C. (2016). The study of nevi in children: Principles learned and implications for melanoma diagnosis. *Journal of the American Academy of Dermatology*, 75(4):813–823. (cited on pages 6, 13, 108, 116, and 186)

Selvaraju, R. R., Das, A., Vedantam, R., Cogswell, M., Parikh, D., and Batra, D. (2016). Grad-cam: Why did you say that? visual explanations from deep networks via gradient-based localization. *CoRR*, abs/1610.02391. (cited on page 46)

Sharif, M., Bhagavatula, S., Bauer, L., and Reiter, M. K. (2019). A general framework for adversarial examples with objectives. *ACM Transactions on Privacy and Security (TOPS)*, 22(3):1–30. (cited on pages 56 and 57)

Simon, M., Gao, Y., Darrell, T., Denzler, J., and Rodner, E. (2017). Generalized orderless pooling performs implicit salient matching. In *Proceedings of the IEEE international conference on computer vision*, pages 4960–4969. (cited on page 49)

Simon, M. and Rodner, E. (2015). Neural activation constellations: Unsupervised part model discovery with convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 1143–1151. (cited on page 51)

Simon, M., Rodner, E., Darrell, T., and Denzler, J. (2018). The whole is more than its parts? from explicit to implicit pose normalization. *IEEE transactions on pattern analysis and machine intelligence*, 42(3):749–763. (cited on page 49)

Simonyan, K., Vedaldi, A., and Zisserman, A. (2013). Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*. (cited on pages 48 and 49)

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*. (cited on page 150)

Stolz, W. and Kunz, M. (2021). Abcd rule — dermoscopedia. `https://dermoscopedia.org/w/index.php?title=ABCD_rule&oldid=15572`. [Online; accessed 1-February-2021]. (cited on pages 1 and 12)

Stomberg, T., Weber, I., Schmitt, M., and Roscher, R. (2021). Jungle-net: Using explainable machine learning to gain new insights into the appearance of wilderness in satellite imagery. *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences*. (cited on page 48)

Strobl, E. V., Zhang, K., and Visweswaran, S. (2019). Approximate kernel-based conditional independence tests for fast non-parametric causal discovery. *Journal of Causal Inference*, 7(1). (cited on pages 40 and 133)

Stutz, D., Hein, M., and Schiele, B. (2019). Disentangling adversarial robustness and generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6976–6987. (cited on pages 9, 55, 56, and 58)

Su, D., Zhang, H., Chen, H., Yi, J., Chen, P.-Y., and Gao, Y. (2018). Is robustness the cost of accuracy?–a comprehensive study on the robustness of 18 deep image classification models. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 631–648. (cited on pages 9, 55, 56, and 59)

Su, J., Vargas, D. V., and Sakurai, K. (2019). One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 23(5):828–841. (cited on pages 52 and 53)

Szegedy, C., Ioffe, S., Vanhoucke, V., and Alemi, A. (2016). Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. *arXiv:1602.07261 [cs]*. arXiv: 1602.07261. (cited on page 109)

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2013). Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*. (cited on pages 9, 10, 49, 50, 52, 54, 56, 57, 59, 61, 62, and 135)

Tabacof, P. and Valle, E. (2016). Exploring the space of adversarial images. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 426–433. IEEE. (cited on pages 55 and 58)

Tan, M. and Le, Q. V. (2020). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *arXiv:1905.11946 [cs, stat]*. arXiv: 1905.11946. (cited on page 110)

Tanay, T. and Griffin, L. (2016). A boundary tilting persepective on the phenomenon of adversarial examples. *arXiv preprint arXiv:1608.07690*. (cited on pages 59, 72, 73, 74, and 187)

Tatu, A., Lauze, F., Nielsen, M., and Kimia, B. (2011). Exploring the representation capabilities of the hog descriptor. In *2011 IEEE international conference on computer vision workshops (ICCV Workshops)*, pages 1410–1417. IEEE. (cited on page 54)

The Cornell Lab of Orniothology (2020). All about birds. `https://www.allaboutbirds.org/guide/White-crowned_Sparrow/species-compare/64980371`. Accessed: 2020-03-01. (cited on page 84)

Thygesen, L. C., Andersen, G. S., and Andersen, H. (2005). A philosophical analysis of the hill criteria. *Journal of Epidemiology & Community Health*, 59(6):512–516. (cited on pages 25 and 27)

Torralba, A. and Efros, A. A. (2011). Unbiased look at dataset bias. In *CVPR 2011*, pages 1521–1528. IEEE. (cited on page 13)

Tramèr, F., Kurakin, A., Papernot, N., Goodfellow, I., Boneh, D., and McDaniel, P. (2017a). Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*. (cited on page 62)

Tramèr, F., Papernot, N., Goodfellow, I., Boneh, D., and McDaniel, P. (2017b). The space of transferable adversarial examples. *arXiv preprint arXiv:1704.03453*. (cited on pages 9, 10, 56, and 58)

Tschandl, P., Codella, N., Akay, B. N., Argenziano, G., Braun, R. P., Cabo, H., Gutman, D., Halpern, A., Helba, B., Hofmann-Wellenhof, R., et al. (2019). Comparison of the accuracy of human readers versus machine-learning algorithms for pigmented skin lesion classification: an open, web-based, international, diagnostic study. *The Lancet Oncology*, 20(7):938–947. (cited on pages 2, 12, 13, 55, and 67)

Tschandl, P., Rinner, C., Apalla, Z., Argenziano, G., Codella, N., Halpern, A., Janda, M., Lallas, A., Longo, C., Malvehy, J., et al. (2020). Human–computer collaboration for skin cancer recognition. *Nature Medicine*, 26(8):1229–1234. (cited on pages 2 and 49)

Tschandl, P., Rosendahl, C., and Kittler, H. (2018). The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions. *Scientific Data*, 5(1):180161. Number: 1 Publisher: Nature Publishing Group. (cited on pages 1, 5, 13, 110, 111, 113, 115, 116, 121, and 155)

Tsipras, D., Santurkar, S., Engstrom, L., Turner, A., and Madry, A. (2018). Robustness may be at odds with accuracy. *arXiv preprint arXiv:1805.12152*. (cited on pages 55, 56, 59, and 66)

Van Horn, G., Mac Aodha, O., Song, Y., Cui, Y., Sun, C., Shepard, A., Adam, H., Perona, P., and Belongie, S. (2018). The inaturalist species classification and detection dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8769–8778. (cited on page 134)

Vapnik, V. (1998). *Statistical learning theory*. Wiley, New York. (cited on pages 30 and 38)

Wang, A., Narayanan, A., and Russakovsky, O. (2020). Revise: A tool for measuring and mitigating bias in visual datasets. In *European Conference on Computer Vision*, pages 733–751. Springer. (cited on pages 13, 14, and 119)

Wang, Z. (2004). Image quality assessment: Form error visibility to structural similarity. *IEEE Trans. Image Process.*, 13(4):604–606. (cited on pages 10, 49, 62, 65, and 67)

Welinder, P., Branson, S., Mita, T., Wah, C., Schroff, F., Belongie, S., and Perona, P. (2010). Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology. (cited on pages 5, 102, 134, 155, and 185)

Wong, E., Schmidt, F., and Kolter, Z. (2019). Wasserstein adversarial examples via projected sinkhorn iterations. In *International Conference on Machine Learning*, pages 6808–6817. PMLR. (cited on page 53)

Wyner, A. D. (1978). A definition of conditional mutual information for arbitrary ensembles. *Information and Control*, 38(1):51–59. (cited on pages 14 and 37)

Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*. (cited on pages 68, 70, and 71)

Young, K., Booth, G., Simpson, B., Dutton, R., and Shrapnel, S. (2019). Deep neural network or dermatologist? In *Interpretability of Machine Intelligence in Medical Image Computing and Multimodal Learning for Clinical Decision Support*, pages 48–55. Springer. (cited on page 108)

Zeiler, M. D. and Fergus, R. (2014). Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer. (cited on pages 9, 44, and 45)

Zhang, B. H., Lemoine, B., and Mitchell, M. (2018a). Mitigating unwanted biases with adversarial learning. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, pages 335–340. (cited on pages 118, 119, 127, 128, 144, 146, 147, 148, 149, and 150)

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. (2016). Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*. (cited on page 68)

Zhang, R., Isola, P., Efros, A. A., Shechtman, E., and Wang, O. (2018b). The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595. (cited on pages 10, 62, 65, and 67)

Zhu, F., Li, H., Ouyang, W., Yu, N., and Wang, X. (2017). Learning spatial regularization with image-level supervisions for multi-label image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5513–5522. (cited on pages 132 and 133)

Zintgraf, L. M., Cohen, T. S., Adel, T., and Welling, M. (2017). Visualizing deep neural network decisions: Prediction difference analysis. *arXiv preprint arXiv:1702.04595*. (cited on page 45)

List of Figures

List of Tables

List of Definitions, Theorems, and Lemmas

Ehrenwörtliche Erklärung

Hiermit erkläre ich ehrenwörtlich, dass mir die am heutigen Tage geltende Promotionsordnung der Fakultät für Mathematik und Informatik an der Friedrich-Schiller-Universität Jena bekannt ist. Ich habe die Dissertation selbstständig angefertigt, habe keine Textabschnitte oder Ergebnisse eines Dritten oder eigenen Prüfungsarbeiten ohne Kennzeichnung übernommen und habe alle von mir benutzten Hilfsmittel, persönliche Mitteilungen und Quellen in meiner Arbeit angegeben. Die Auswahl und Auswertung sämtlichen Materials geschah eigenständig. Insbesondere habe ich keine Hilfe eines Promotionsberaters in Anspruch genommen. Auch haben Dritte weder unmittelbar noch mittelbar geldwerte Leistungen von mir für Arbeiten erhalten, die im Zusammenhang mit dem Inhalt der vorgelegten Dissertation stehen.

Des Weiteren erkläre ich, dass ich die vorliegende Arbeit noch nicht als Prüfungsarbeit für eine staatliche oder andere wissenschaftliche Prüfung, sowie keine identische, eine in wesentlichen Teilen ähnliche oder eine andere Abhandlung bei einer anderen Hochschule als Dissertation eingereicht habe.

Jena, den Christian Reimers