Friedrich-Schiller-Universität Jena
Fakultät für Mathematik und Informatik
Lehrstuhl für Digitale Bildverarbeitung

Semantic Knowledge Integration for Learning from Semantically Imprecise Data

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 Clemens-Alexander Brust, M. Sc.
geboren am 1. Mai 1991 in Hannover

Gutachter

1. Prof. Dr.-Ing. Joachim Denzler
   Friedrich-Schiller-Universität Jena,
2. Prof. Dr.-Ing. Patrick Mäder
   Technische Universität Ilmenau

Tag der Einreichung	30. August 2021
Tag der öffentlichen Verteidigung	7. April 2022

Abstract

Low availability of labeled training data often poses a fundamental limit to the accuracy of computer vision applications using machine learning methods. While these methods are improved continuously, *e.g.*, through better neural network architectures, there cannot be a single methodical change that increases the accuracy on all possible tasks. This statement, known as the *no free lunch* theorem, suggests that we should consider aspects of machine learning other than learning algorithms for opportunities to escape the limits set by the available training data.

In this thesis, we focus on two main aspects, namely the nature of the training data, where we introduce structure into the label set using concept hierarchies, and the learning paradigm, which we change in accordance with requirements of real-world applications as opposed to more academic setups.

Concept hierarchies represent semantic relations, which are sets of statements such as "a bird is an animal". We propose a hierarchical classifier to integrate this *domain knowledge* in a pre-existing task, thereby increasing the information the classifier has access to. While the hierarchy's leaf nodes correspond to the original set of classes, the inner nodes are "new" concepts that do not exist in the original training data.

However, we pose that such *imprecise* labels are valuable and should occur naturally, *e.g.*, as an annotator's way of expressing their uncertainty. Furthermore, the increased number of concepts leads to more possible search terms when assembling a web-crawled dataset or using an image search. We propose CHILLAX, a method that learns from semantically imprecise training data, while still offering precise predictions to integrate seamlessly into a pre-existing application.

The common learning paradigm of "waterfall" learning, where training images are first collected, then annotated and finally used for learning, does not align well with real-world applications. When machine learning methods are used to assist in research projects, *e.g.*, camera trap image analysis, the training data is not fully available from the beginning. Instead, it is slowly collected over time, and annotation resources are also rarely available all at once. The *lifelong learning* framework proposes a learning cycle which consists of repeated selections of unlabeled images, annotations, and model updates.

We propose an active learning method to intelligently select images for annotation in a lifelong object detection task, *e.g.*, biodiversity monitoring. To increase the speed of model updates, we adapt an incremental learning method to the object detector, eliminating the need for expensive re-training from scratch. We further present a working implementation of a full lifelong learning system used in a real-world biodiversity monitoring project.

Zusammenfassung

Die geringe Verfügbarkeit annotierter Trainingsdaten begrenzt häufig die mögliche Genauigkeit von Anwendungen der Bildverarbeitung auf der Grundlage von maschinellen Lernverfahren. Obwohl diese Methoden stetig verbessert werden, z.B. durch bessere neuronale Netzarchitekturen, kann es keine einzelne methodische Veränderung geben, die die Genauigkeit auf allen möglichen Aufgabenstellungen erhöht. Aufgrund dieses *No Free Lunch*-Theorems sollten wir uns auf andere Aspekte des maschinellen Lernens abseits von Lernalgorithmen konzentrieren, um die Begrenzungen durch wenig verfügbare Trainingsdaten zu umgehen.

Die zwei Schwerpunkte dieser Arbeit betreffen einerseits die Beschaffenheit der Trainingsdaten, deren Klassenmenge wir durch Konzepthierarchien Struktur verleihen, und andererseits das Lernparadigma, das wir anpassen, um auf die Bedürfnisse von Anwendungen in der echten Welt Rücksicht zu nehmen.

Konzepthierarchien sind Repräsentationen semantischer Relationen, also Mengen von Aussagen wie "ein Vogel ist ein Tier." Wir stellen einen hierarchischen Klassifikator vor, der dieses *Domänenwissen* in eine bestehende Aufgabe integriert, sodass dem Klassifikator mehr Information zugänglich ist. Die Blattknoten dieser Hierarchie entsprechen der ursprünglichen Klassenmenge, während die inneren Knoten "neue" Konzepte sind und in den ursprünglichen Trainingsdaten nicht direkt vorkommen.

Solche *unpräzisen* Annotationen sind unserer Anschauung nach allerdings wertvoll, und sollten selbstverständlich vorkommen, z.B. damit Annotierende ihre Unsicherheit ausdrücken können. Außerdem gibt es durch die höhere Gesamtzahl an Konzepten mehr mögliche Schlagworte, um Trainingsdaten durch einen Datensatz aus Bildersuche zu erweitern. Wir stellen CHILLAX vor, eine Methode, die aus semantisch unpräzisen Trainingsdaten lernt, aber dennoch präzise Vorhersagen liefert und damit nahtlos in eine bestehende Anwendung integriert werden kann.

Das typische Paradigma des "Wasserfall"-Lernens, wobei Trainingsbilder erst gesammelt, anschließend annotiert und letztendlich zum Lernen verwendet werden, ist nicht gut auf Anwendungen in der echten Welt ausgerichtet. Wenn Forschungsprojekte maschinelle Lernverfahren einsetzen, z.B. bei der Analyse von Kamerafallenbildern, sind die Trainingsdaten nicht von Anfang an voll verfügbar. Sie werden stattdessen allmählich aufgenommen, und auch die personellen Ressourcen zur Annotation können nicht alle auf einmal abgerufen werden. Das Framework des *lebenslangen Lernens* schlägt einen Kreislauf vor, der aus wiederholten Auswahlen von nicht annotierten Bildern, Annotationsanfragen und Modellaktualisierungen besteht.

Wir stellen eine Methode des aktiven Lernens vor, die Bilder für eine Objektdetektionsaufgabe, z.B. in der Biodiversitätsüberwachung, intelligent zur Annotation auswählt. Um das Modell schneller zu aktualisieren, passen wir ein inkrementelles Lernverfahren an den Detektor an. Zudem stellen wir eine vollständige Implementierung eines lebenslangen Lernsystems vor, die in einem echten Forschungsprojekt zur Biodiversitätsüberwachung verwendet wird.

Contents

1. Introduction **1**
1.1. Requirements of Real-World Applications 2
1.2. Overview of Contributions 3
1.3. Remainder of this Thesis 5

2. Machine Learning in the Past and Present **7**
2.1. Machine Learning Foundations 7
    2.1.1. Statistical Learning Theory 7
    2.1.2. Linear Model 13
    2.1.3. Gradient-Based Optimization 14
    2.1.4. Supervised Learning and Alternatives 16
2.2. Model Selection and Complexity 17
    2.2.1. Regularization 18
    2.2.2. Validation 19
2.3. Machine Learning for Computer Vision 21
    2.3.1. Images 21
    2.3.2. Computer Vision Tasks 22
    2.3.3. Challenges 23
    2.3.4. Feature Extraction and Invariances 25
2.4. Deep Learning 26
    2.4.1. Artificial Neural Networks 26
    2.4.2. Convolutional Neural Networks 29
    2.4.3. Implementation Details 31

3. Concept Hierarchies as Semantic Knowledge **33**
3.1. Formal Introduction 33
    3.1.1. Concepts 34
    3.1.2. Relations 35
3.2. Concept Hierarchies 37
    3.2.1. Concept Hierarchies from Relations 38
    3.2.2. Concept Hierarchies as Graphs 38
3.3. Semantic Knowledge Bases 40
    3.3.1. Lexical Databases 41
    3.3.2. Visual-Semantic Datasets 42
    3.3.3. Domain-Specific Knowledge Bases and the Semantic Web . . . 44

4. Literature Overview **47**
4.1. Semantic Knowledge Integration From Concept Hierarchies 47
    4.1.1. Model Structure 47
    4.1.2. Embeddings 50

4.1.3. Metric Learning . . . 51
4.1.4. Non-Semantic Hierarchies . . . 52
4.1.5. Analysis and Exploratory Works . . . 53
4.2. Problem Formulations . . . 53
4.2.1. Imprecise Data . . . 53
4.2.2. Fine-Grained and Large-Scale Classification . . . 54
4.2.3. Image Retrieval . . . 55
4.2.4. Alternatives to Supervised Learning . . . 55
4.2.5. Natural Language Processing . . . 57

5. Semantic Knowledge Integration and Learning from Imprecise Data 59
5.1. Imprecise Data . . . 59
5.1.1. Measuring and Modeling Imprecision using Concept Hierarchies 60
5.1.2. Mapping Concepts to Datasets . . . 64
5.2. Knowledge Integration by Hierarchical Classification . . . 67
5.2.1. Internal Representation of Labels . . . 68
5.2.2. Introducing Assumptions . . . 69
5.2.3. Hypothesis . . . 70
5.2.4. Deep Learning Implementation . . . 72
5.3. Extrapolating Imprecise Data . . . 74
5.3.1. Problem Formulation . . . 74
5.3.2. From Hierarchical Classifier to CHILLAX . . . 76
5.3.3. Self-Supervised Approach . . . 77

6. Experiments on Knowledge Integration and Imprecise Data 83
6.1. Evaluation . . . 83
6.1.1. Classification . . . 83
6.1.2. Hierarchical Classification . . . 85
6.2. Experimental Setup . . . 86
6.2.1. Datasets . . . 86
6.2.2. Deep Neural Networks . . . 88
6.2.3. Data Processing . . . 89
6.3. Knowledge Integration on Benchmark Datasets . . . 90
6.3.1. Setup . . . 91
6.3.2. Results . . . 92
6.4. Imprecise Data from Benchmark Datasets . . . 95
6.4.1. Simulating Label Noise . . . 96
6.4.2. Supervised CHILLAX . . . 97
6.4.3. Self-Supervised CHILLAX . . . 102
6.5. Imprecise Data from Webly Supervision: Moths . . . 108
6.5.1. Setup . . . 108
6.5.2. Results . . . 111
6.6. Pitfalls and Dangers . . . 113
6.6.1. Visual-Semantic Correspondence . . . 113
6.6.2. Harmful Hierarchies . . . 116
6.6.3. Imprecision in Real-World Data . . . 120

6.7. Summary and Discussion 122
6.7.1. Benchmark Datasets 123
6.7.2. Webly Supervision 125
6.7.3. Pitfalls and Dangers 125
6.7.4. Omissions and Limitations 127

7. Experimental Outlook Towards Lifelong Learning 129
7.1. A Brief Introduction to Lifelong Learning 129
7.1.1. Active Learning 130
7.1.2. Incremental Learning 131
7.1.3. The Lifelong Learning Cycle 131
7.2. Active and Incremental Learning for Object Detection 132
7.2.1. Active Learning for Object Detection 132
7.2.2. Incremental Learning of YOLO 134
7.2.3. Experiments 134
7.2.4. Summary and Discussion 136
7.3. Weakly Supervised Lifelong Learning for Object Detection 137
7.3.1. Interaction Model 137
7.3.2. Camera Traps 138
7.3.3. Experiments 138
7.3.4. Carpe Diem Annotation Tool 139
7.4. Label-Efficient Gorilla Re-Identification 140
7.4.1. Re-Identification by Face Detection 141
7.4.2. Experiments 141
7.4.3. Discussion 143
7.5. Lifelong Learning with Concept Hierarchies 144

8. Conclusion 145
8.1. Summary 145
8.2. Future Work 148

A. Appendix 151

Notation 157

List of Figures 159

List of Tables 161

Acronyms 163

Glossary 165

References 169

List of Own Publications 191

Ehrenwörtliche Erklärung 195

1. Introduction

There is a global biodiversity crisis (*cf.* Cardinale et al. 2012; Vié, Hilton-Taylor, and Stuart 2009; Vogel 2017) which poses a severe threat to our planet's habitability. Its main drivers include global phenomena such as climate change (*cf.* Thomas et al. 2004) and local effects, *e.g.*, habitat destruction (*cf.* Tilman et al. 1994). Visual monitoring using camera traps (see section 7.3.2) is a key ingredient to shaping an appropriate response and guiding policy.

At the same time, the raw images are not informative on their own. They require analysis to estimate occupancy, abundance and other indicators, and the scale of data generated from camera trap operations makes individual viewing of images by biodiversity researchers infeasible. Automated processing using computer vision and machine learning is the only practical option for long-term, continuous and non-invasive biodiversity monitoring.

However, modern and highly accurate methods such as convolutional neural networks depend on large quantities of training data, which can be cost-prohibitive considering the expertise required in order to produce reliable annotations. And biodiversity research is only one of the applications of computer vision and machine learning where data availability poses a fundamental limit to performance, and therefore usefulness. General methodical improvements such as better neural network architectures, loss functions etc. are not guaranteed to translate to any specific problem, which is the subject of the *no free lunch* theorem (see section 2.1.1.5). Hence, we should explore alternative solutions. In this thesis, we explore two main "escape routes" out of such a situation, which simultaneously address further specific needs of real-world applications (see section 1.1).

The first escape route concerns the training data. We make use of readily available concept hierarchies, *e.g.*, the biological taxonomy, which constitute domain knowledge that can be used in addition to the scarce training data. And as stated in Shalev-Shwartz and Ben-David (2014, p. 40), "we can escape the hazards foreseen by the No-Free-Lunch theorem by using our prior knowledge about a specific learning task". Accordingly, we leverage concept hierarchies using our probabilistic hierarchical classifier, integrating the domain knowledge contained therein for increased accuracy without more requiring more training data.

However, that is only a first step. With concept hierarchies and our classifier, we not only improve performance, but also gain access to imprecise training data. Semantically imprecise data can occur when annotators are only certain of their label up to a some level of precision, *e.g.*, the family level, while their actual task is to annotate species. If annotators were then forced to annotate at the species level, they could introduce errors or refuse to annotate examples unless they are certain enough. However, if they are allowed to select a less precise label, *e.g.*, a genus or a family, they can express their knowledge without loss of information or errors. Such a flexible process is especially relevant for citizen science projects involving people of varying expertise and enables a conscious trade-off between quality and quantity of labels. Furthermore, when crawling the web for additional data, there is a higher number of search terms. We propose methods for learning from such imprecise data as well as probabilistic models of annotator precision.

The second escape route concerns the learning paradigm itself. We acknowledge that the “waterfall” model of machine learning (see Data Science Process Alliance 2021), where a model is trained once and then used indefinitely, does not align well with real-world applications. With camera traps for example, new data is produced continuously, and distributing annotation sessions over time aligns them more closely with working hours in a project. We propose active learning methods to intelligently decide which unlabeled images are valuable enough to annotate and integrate them with a lifelong learning system. Moreover, we consider the human-machine interaction aspect and investigate a combination of our methods with a fast annotation process that produces weak labels for object detection.

1.1. Requirements of Real-World Applications

In this section, we explore properties of machine learning and computer vision applications in the real world, which are not reflected in a typical academic machine learning environment. The latter is discussed in terms of waterfall learning in section 7.1.

The first property is the requirement of representative training data (see section 2.1.1.7) sampled from an environment distribution. Representativeness implicitly assumes that all annotations are “correct”, at least in terms of the distribution. However, human annotators are not perfect (*cf*. Russakovsky et al. 2015). Furthermore, they are not all equally knowledgeable (*cf*. Chang et al. 2021), meaning that each annotator should be characterized by their own distribution. It is therefore challenging to obtain completely representative training data using a group of human annotators, and also to decide which samples represent the “real” distribution that should be learned.

Similar concerns can be raised w.r.t. collecting the images, not only the labels. Assuming a fixed environment distribution discounts the prospect of changes over time. Whereas in reality, concepts naturally “drift” as time passes. And in a continuous application such as camera trap analysis, the images simply are not available in full at the beginning of a project. Instead, they are recorded over time, and older images lose relevance as concepts and the environment change. Hence, a fixed distribution which is sampled once to generate training data is not always a reasonable representation of the real world.

If the environment distribution is allowed to change over time, and images are represented by a continuous data stream, the annotation process has to be adapted as well. As old images become irrelevant, so do the respective labels. Regular annotation sessions have to be coordinated with the availability of new images to capture the changing distribution. And as the training data changes, models derived from the data require constant re-training as well.

1.2. Overview of Contributions

This section gives on overview of our contributions in this thesis, which mainly consist of methods that address the aforementioned requirements, and analyses of failure cases as well as verification of assumptions made by our methods.

Hierarchical Classification Our hierarchical classifier, as proposed in section 5.2 and published in Brust and Denzler (2019a), integrates domain knowledge and serves as the foundation for the following methods concerning imprecise data. From the simple assumptions (see section 5.2.2) of subsumption informed by a hierarchy, *e.g.*, "a bird has to be an animal", and a closed world, *i.e.*, "everything is an object", we derive a probabilistic model that relates concepts. We use the term *concept* to differentiate from classes, which are always assumed to be mutually exclusive (whereas `bird` and `animal` are not). To integrate the probabilistic model into a deep learning setup, we transform it into a label encoding and a loss function in section 5.2.4. While our hierarchical classifier is the foundation for the following contributions, it also has merit on its own as a way of leveraging domain knowledge to improve accuracy.

Learning from Imprecise Data With imprecise data, we allow any concept in a hierarchy as a label to utilize the individual expertise of each annotator. CHILLAX (**c**lass **h**ierarchies for **i**mprecise **l**abel **l**earning and **a**nnotation e**x**trapolation) is proposed in section 5.3.2 and published in Brust, Barz, and Denzler (2021b). The method is based on the aforementioned hierarchical classifier. It interprets imprecise labels such as `animal` as uncertain w.r.t. subsumed concepts, meaning "this is certainly an animal, but I'm unsure which". CHILLAX learns from such labels, but makes precise predictions at the same time, *i.e.*, it predicts only leaf nodes in the respective hierarchy. With this capability, which we call *extrapolation*, we address the different levels of expertise in individual annotators as described in section 1.1 and allow for non-representative training data w.r.t. the label distributions. We further propose a self-supervised strategy in section 5.3.3 which enables CHILLAX to learn from effectively unlabeled images.

Lifelong Learning In chapter 7, we consider the specific needs of applications where the environment distribution changes over time using the framework of lifelong learning. An active learning method for object detection tasks is proposed in section 7.2.1 and published in Brust, Käding, and Denzler (2019). Based on uncertainty heuristics, it intelligently selects unlabeled images for annotation to maximize the utility given a constrained labeling budget. Moreover, we propose a modification to YOLO (Redmon et al. 2016) that enables incremental learning in section 7.2.2, including the addition of new concepts as they are discovered. Incremental learning significantly reduces training time as new data becomes available by eliminating the need for re-training.

Real-World Applications We validate our methods not only on benchmark data, but also apply them to three real-world biodiversity research applications. The first application concerns moth species classification and is detailed in section 6.5. We map the species in the dataset to the WikiSpecies database (see section 3.3.3.1) to obtain a complete taxonomy. For each taxon, including genera, families etc., we download images from Flickr and learn the resulting imprecise data using CHILLAX.

The second application, described in section 7.3 and published in Brust, Käding, and Denzler (2020), highlights our contributions to lifelong learning in automated camera trap setup. We implement and validate a complete lifelong learning system that supports intelligent selection of unlabeled examples, fast incremental learning, and can accept new annotations at any time. The system is developed specifically for ease of use and the graphical user interface implementation is published separately in Brust, Barz, and Denzler (2021a).

Our third application is detailed in section 7.4 and published in Brust et al. (2017). It considers heuristics to use the combination of face detection and classification for individual re-identification of gorillas. The heuristics are necessary to map the single label per image to potentially multiple detected faces correctly. Annotators only have to provide identification, while the face is detected automatically. This combination of methods and the separation of concerns allows for high sample efficiency and fast annotation times.

Analyses We utilize concept hierarchies for their domain knowledge and to introduce structure to a set of labels such that annotators can produce imprecise labels if they are uncertain. However, both use cases assume that the hierarchy is correct and that its semantic relation aligns to the visual properties of the corresponding dataset. We conduct a study, detailed in section 6.6.1 and published in Brust and Denzler (2019b), where we actually measure the visual-semantic correspondence on a benchmark dataset to validate the latter assumption. In section 6.6.2, we consider faulty hierarchies and determine the effects of "swapped" relations on the accuracy of our hierarchical classifier. We further propose a synthetic dataset with perfect visual-semantic correspondence in section 6.6.2.2 for additional insight by comparing it to real-world data.

Since part of our experiments concerning imprecise data rely on models of annotator precision, we conduct an investigation using Flickr to validate these models, which is published in Brust, Barz, and Denzler (2021b) and described in section 6.6.3. We determine precision distributions over the image title, caption and further metadata, using our algorithm for mapping arbitrary text to WordNet synsets, which we propose in section 5.1.2.

Open-Source Software We publish the source code of implementations of our methods wherever possible, and bundle them in reusable projects to encourage further development and replication. The main software product of this thesis is CHIA[1], which implements our hierarchical classifier (see section 5.2), connections to knowledge bases (see section 3.3) as well as benchmark datasets (see section 6.2.1). CHILLAX (see section 5.3.2) is published separately[2], including the self-supervised methods discussed in section 5.3.3.

---

[1] `https://github.com/cvjena/chia`

[2] `https://git.inf-cv.uni-jena.de/brust/chillax`

Our contributions in chapter 7 are implemented as part of our deep learning framework CN24[3], which uses OpenCL for hardware acceleration. We also publish the source code of Carpe Diem[4], our graphical user interface for lifelong learning of object detection tasks, which we discuss in detail in section 7.3.4.

All aforementioned projects are published under a 3-clause BSD license. This license allows for commercial and private use, free distribution as well as modification. It admits neither warranty nor liability.

1.3. Remainder of this Thesis

In the following, we briefly describe the structure of the thesis and the content of the chapters following this one.

Chapter 2 introduces foundational concepts in machine learning. We define the individual components of a machine learning system and make statements w.r.t. learnability in section 2.1, where we also discuss basic methods and algorithms. In section 2.2, we acknowledge overfitting as a general problem and detail validation methods to tackle it. The tasks and challenges associated with computer vision for machine learning are described in section 2.3. It further introduces feature representation, and we discuss deep learning as an alternative to hand-engineered features in section 2.4.

The third chapter concerns concept hierarchies as one representation of semantic knowledge. We give a formal introduction to concepts and semantic relations in section 3.1. From these relations, we derive hierarchies and discuss their graph representations in section 3.2. Finally, section 3.3 explores a large selection of sources for concept hierarchies including linguistic, biological and medical subjects.

Chapter 4 lists previous work related to our own, where we focus on two main areas. First, we discuss methods that involve concept hierarchies to integrate their knowledge directly or indirectly in section 4.1. Second, we consider problem formulations that incorporate quantitatively or qualitatively deficient training data, as well as methods to solve these problems in section 4.2, where we relate these tasks to learning from imprecise data.

Chapter 5 bundles our methodical contributions. In section 5.1, we first define imprecise data formally and propose a probabilistic model of imprecision for various categories of annotators. We also propose an algorithm to automatically match concept hierarchies to existing datasets. In section 5.2, we construct our hierarchical classifier from first principles and develop a deep learning implementation of its probabilistic model. The classifier is modified in section 5.3 to build CHILLAX, enabling learning from imprecise data. Furthermore, we analyze drawbacks of CHILLAX and propose self-supervised methods to counteract them.

The aforementioned methods concerning the "first escape route" are validated empirically in chapter 6. We first introduce evaluation criteria in section 6.1 and identify common elements of our experimental setups in section 6.2. Our hierarchical classifier is tested for its capability of integrating domain knowledge on small- and large-scale benchmark datasets in section 6.3. We then validate learning from

---

[3] `https://github.com/cvjena/cn24`

[4] `https://git.inf-cv.uni-jena.de/LifelongLearning/carpediem`

imprecise data using CHILLAX on large-scale benchmark data and compare it to a competitor in section 6.4. In section 6.5, a real-world biodiversity research application is improved using imprecise data crawled from the web. The increased complexity of hierarchical classification also introduces new potentials for errors, which we analyze in a detailed manner in section 6.6. Finally, section 6.7 summarizes the results and offers explanations and discussion.

We discuss our contributions to the "second escape route" in chapter 7, starting with an introduction to the idea of lifelong learning in section 7.1. An active learning method that intelligently selects unlabeled images for annotation with bounding boxes is presented in section 7.2. We then introduce a weak annotation mechanism to the system and evaluate it in a continuous biodiversity monitoring scenario in section 7.3. In section 7.4, we discuss a further biodiversity application involving monitoring and re-identification of individuals. As an outlook, we propose a combination of both escape routes in the form of an active learning method for hierarchical classification in section 7.5.

The thesis concludes with chapter 8, which summarizes our findings in section 8.1 and gives an overview of promising future research directions in section 8.2.

2. Machine Learning in the Past and Present

This chapter introduces the concepts in machine learning that are relevant to this work. There are many ways to approach an introduction to machine learning which are equally legitimate. For example, one can look at data as a central topic, or start with certain models. A historical view can also be enlightening. We choose the angle of statistical learning theory (Vapnik 1998; Vapnik and Chervonenkis 1971) as interpreted in Shalev-Shwartz and Ben-David (2014), which has its foundations in probability theory and statistics. This approach integrates well with this thesis because we formulate our models in chapter 5 in a probabilistic framework.

Sections 2.1 and 2.2 introduce machine learning in an application-agnostic way. We build on this foundation in section 2.3, where the specific needs of computer vision tasks are characterized. In section 2.4, we describe the more recent idea of deep learning.

2.1. Machine Learning Foundations

Before considering machine learning, one needs to discuss the definition of learning in general. The Oxford English Dictionary defines learning as "to acquire knowledge of (a subject) or skill in (an art, etc.) as a result of study, experience, or teaching" (Oxford English Dictionary 2020). An alternative definition given by Washburne (1936) is "an increase, through experience, of problem-solving ability."

Machine learning follows from these definitions by replacing the "learner" with a machine. We usually stress the latter definition, as the problem-solving ability is a desirable property. The realization of knowledge in machine learning is less important for any given problem. It is rather a consequence of which specific method is used, including some that specify knowledge explicitly (see section 4.1).

In this section, we first build a theoretical foundation in section 2.1.1. We then introduce implementations in sections 2.1.2 and 2.1.3 and discuss common problem formulations in section 2.1.4.

2.1.1. Statistical Learning Theory

We introduce the central concepts in machine learning using the framework of *statistical learning theory* (Vapnik 1998; Vapnik and Chervonenkis 1971). Initially, we consider a *prediction task*. The goal is to infer the state of one random variable, which is hidden, from the observation of another. This is only possible if the two variables are dependent in some way.

A common example is that of spam mail. Whether an e-mail is spam or not cannot be observed by the recipient — the intent is only known to the sender. However, the content of the e-mail can be used to *classify* it. We assume that there exist *features*, *e.g.*, the presence of certain keywords (`viagra`, `bitcoin`), that are related to the intent.

If such a relationship exists, we can formulate rules. For example: "if the mail contains `prince` and `inheritance`, it is spam". While such simple rules can be generated ad-hoc, this is not feasible for more complicated problems, *e.g.*, image classification (see section 2.3.2.1). Instead, we *learn* the rules from *data*, which are joint samples of both the hidden and visible random variables. Through *generalization*, these rules also apply to previously unseen observations. In the following, we formalize this process and the components of a learning system.

2.1.1.1. Ingredients

We consider the main ingredients of a machine learning and their formal definitions in the context of a prediction task. Our terminology and derivation follows that of Shalev-Shwartz and Ben-David (2014), which presents a contemporary take on statistical learning theory.

First, we specify the data on which our prediction system operates. From an application perspective, a predictor gets an input (*e.g.*, the content of the mail) and returns an output (whether it is spam or not). The corresponding ranges of input and output are formalized as follows:

Definition 2.1 (domain set)**.** The *domain set* $\mathcal{X}$ is the set of possible "inputs" to a predictor, also termed *domain points*. Elements of this set are to be labeled.

Definition 2.2 (label set)**.** The *label set* $\mathcal{Y}$ is the set of possible "outputs" of a predictor. These are the *labels* associated with the elements of the domain set. Depending on the structure of the label set, we distinguish different tasks:

- *Classification* if $\mathcal{Y}$ is a set of mutually exclusive classes, *e.g.*, $\mathcal{Y} = \{\texttt{dog}, \texttt{cat}, \texttt{car}\}$.
- *Regression* if $\mathcal{Y}$ is continuous, *e.g.*, $\mathcal{Y} = \mathbb{R}$.

The association between domain points and labels is often determined by humans in an *annotation* process. The human component is implied by using the word label, but labels can also be a physical measurement, or any other random variable that cannot be observed in the future, and thus has to be predicted. In any case, the results of an annotation or measurement process are combined to form a set of data:

Definition 2.3 (training data)**.** The *training data* S is a sequence of m pairs in $\mathcal{X} \times \mathcal{Y}$. It is typically indexed, such that $S = \big((x_1, y_1), \ldots, (x_m, y_m)\big)$. While the order of the pairs should not be of consequence, the training data is not defined as a set because it could contain the same pair multiple times.

To make a prediction, we use a function that maps from observations in the domain set to predictions in the label set. Formally, this predictor function is defined as:

Definition 2.4 (hypothesis)**.** A *hypothesis* is a function $h : \mathcal{X} \to \mathcal{Y}$ from the domain set to the label set.

We use the terms hypothesis, model, and predictor interchangeably. One ingredient still missing is the actual learning step:

Definition 2.5 (learner)**.** A *learner* $\mathcal{A}(S)$ is an algorithm that takes training data and produces a hypothesis.

A complete learning system realizes all of the aforementioned components. While the domain set and label set are specific to the task or application at hand, there are many general implementations of learners, which we discuss in the following sections.

2.1.1.2. Probability Distributions

Before we explore the implementations and details of learners, we further formalize how our training data is generated. For now, we assume that there exists a function that can assign the "correct" label to every element of the domain set $\mathcal{X}$ (*cf.* ibid.).

Definition 2.6 (target function)**.** The *target function* is a function $f : \mathcal{X} \to \mathcal{Y}$ from the domain set to the label set. It always assigns the "correct" label $y = f(x)$

Ideally, a human annotator perfectly executes this target function. A deterministic functional relationship between domain set and label set is a very strong assumption that often does not hold in the real world. We discuss a relaxation of this assumption in section 2.1.1.7. We further assume that there exists a distribution to sample training data from:

Definition 2.7 (environment distribution)**.** The *environment distribution* $\mathcal{D}$ over the domain set $\mathcal{X}$ describes the environment. Training data is generated by sampling from elements from $\mathcal{X}$ according to $\mathcal{D}$. The target function is then queried to label each sample, resulting in a sequence $\big((x_1, y_1), \ldots, (x_m, y_m)\big) = \big((x_1, f(x_1)), \ldots, (x_m, f(x_m))\big)$.

2.1.1.3. (Empirical) Risk Minimization

The name "target function" already suggests that it is something to strive for. In fact, the main goal for a learner is to produce a hypothesis that matches the target function as closely as possible.

Definition 2.8 (risk)**.** Given the environment distribution $\mathcal{D}$ and target function f, we define the *risk* $L_{\mathcal{D},f}(h)$ of a hypothesis h as:

$$L_{\mathcal{D},f}(h) = \mathbb{P}_{x \sim \mathcal{D}}\big[h(x) \neq f(x)\big].$$

In other words, the risk is the probability (over the whole domain set) of the hypothesis and the target function disagreeing.

A successful learner should then minimize the risk $L_{\mathcal{D},f}$ to achieve its goal. However, it cannot do so directly because both $\mathcal{D}$ and f are unknown to the learner. Instead, it has to rely solely on the training data S. We can derive:

Definition 2.9 (empirical risk)**.** Given training data $S = \big((x_1, y_1), \ldots, (x_m, y_m)\big)$, the *empirical risk* $L_S(h)$ of a hypothesis h is:

$$L_S(h) = \frac{1}{m} \sum_{i=1}^{m} \mathbb{1}[h(x_i) \neq y_i].$$

Empirical risk can be optimized directly. This type of learning is denoted *empirical risk minimization (ERM)*. It is the basis for all following methods and algorithms. However, minimizing empirical risk does not necessarily minimize risk. There are additional assumptions necessary to relate the two, which are discussed in the following, and specifically section 2.1.1.5. The relationship between risk and empirical risk is detailed further in section 2.2.

2.1.1.4. A Trivial Learner

The formalisms outlined up to this point are technically sufficient to construct a complete learning system. ERM naturally induces a learner as an optimization problem:

$$\mathcal{A}_{\text{ERM}}(S) = \text{ERM}(S) = \arg\min_{h} L_S(h). \tag{2.1}$$

Note that the search space for h is intentionally vague. This learner allows for every possible hypothesis. In fact, there are infinitely many trivial solutions with $L_S = 0$. However, we define the empirical risk as a proxy for the risk only because our learner lacks access to the environment distribution $\mathcal{D}$ and target function f. And while $\mathcal{A}_{\text{ERM}}(S)$ may be perfectly equal to f on the subset of $\mathcal{X}$ supported by S, there are no constraints on the rest of the domain. In this framework, risk and empirical risk are not necessarily related, which may seem catastrophic for ERM since our main goal is the minimization of risk. However, we can relate risk and empirical risk by introducing assumptions that constrain the search space, *e.g.*, from domain knowledge. Such an assumption is referred to as an *inductive bias*. In the following, we show how bounds on the risk can be obtained from applying such biases.

2.1.1.5. Inductive Biases

The first and most important assumption to make is w.r.t. the training data S. We require the elements x_i to be *i.i.d.* samples of $\mathcal{D}$, meaning independent and identically distributed (*cf.* Shalev-Shwartz and Ben-David 2014, p. 18). The *i.i.d.* assumption is fundamental to machine learning. By convention, it is only relaxed on purpose, *e.g.*, to learn with time series (Bishop 2008, p. 605).

Our next step is to constrain the allowed hypotheses to some class $\mathcal{H}$. This step is an opportunity to apply prior knowledge to the problem. Without prior knowledge, selecting a learner that is generally better than another is impossible, which is also called the *no free lunch theorem* (*cf.* Goodfellow, Bengio, and Courville 2016, p. 116).

For the next section, we temporarily add another constraint to our problem. We assume that there exists a hypothesis $h^* \in \mathcal{H}$, such that $L_{\mathcal{D},f}(h^*) = 0$. This is the *realizability assumption* (*cf.* Shalev-Shwartz and Ben-David 2014, p. 17).

2.1.1.6. PAC Learnability

Not all hypothesis classes are created equal. In the following, we consider definitions of learnability based on the assumptions made previously.

The first is probably approximately correct (PAC) learnability (Valiant 1984). We follow the definition given by Shalev-Shwartz and Ben-David (2014, p. 22).

Definition 2.10 (PAC learnable)**.** A given hypothesis class $\mathcal{H}$ is PAC learnable, if a function $m_{\mathcal{H}} : (0,1)^2 \rightarrow \mathbb{N}$ and a learner $\mathcal{A}$ exist, such that for every:

- $\epsilon \in (0,1)$ (*accuracy*) and $\delta \in (0,1)$ (*confidence*),
- environment distribution $\mathcal{D}$ over $\mathcal{X}$,
- target function $f : \mathcal{X} \rightarrow \{0,1\}$,

if both the realizability assumption and the *i.i.d.* assumption on the training data S hold, then evaluating $\mathcal{A}(S)$ with $|S| = m \geq m_{\mathcal{H}}(\epsilon, \delta)$ results in a hypothesis h, where with probability of at least $1 - \delta$, the risk is:

$$L_{\mathcal{D},f}(h) \leq \epsilon.$$

If we can find such a PAC learnable class of hypotheses, we finally have a bound on the risk itself. However, this definition is only valid for binary classification tasks, where there are exactly two labels. Furthermore, in order for a PAC learnable $\mathcal{H}$ to exist for such a task, a function $m_{\mathcal{H}}$ has to exist as well. This function is also termed *sample complexity*. For finite hypothesis classes, such a function always exists (*cf.* ibid., p. 23):

$$m_{\mathcal{H}}(\epsilon, \delta) \leq \lceil \frac{\log \frac{|\mathcal{H}|}{\delta}}{\epsilon} \rceil$$

2.1.1.7. From Target Function to Loss Function

For practical purposes, we are required to relax our previous assumptions slightly, even if it should result in a worse lower bound for the risk.

Target Function First, we remove the target function f in favor of an environment distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$. The corresponding risk is (*cf.* ibid., p. 24):

$$L_{\mathcal{D}}(h) = \mathbb{P}_{(x,y) \sim \mathcal{D}}\left[h(x) \neq y\right]. \tag{2.2}$$

The training data is sampled from $\mathcal{D}$ in pairs (x, y). While the target function is deterministic, this formulation in contrast allows for situations where a domain point is labeled ambiguously. This means that the lower bound for $L_{\mathcal{D}}$ may no longer be zero regardless of $\mathcal{H}$, because the hypothesis is still deterministic. Consequently, we also remove the realizability assumption since it is unreasonably strong in this context (*cf.* ibid., pp. 23 sq.).

Loss Function Since the target function no longer exists in our setting, its range no longer affects the label set. The range of f as given in definition 2.10 implies $\mathcal{Y} = \{0,1\}$. We remove this restriction to binary classification in favor of an unrestricted label set $\mathcal{Y}$. However, this relaxation has consequences for the risk. For continuous $\mathcal{Y}$, *e.g.*, $\mathcal{Y} = \mathbb{R}$, the inequality in eq. (2.2) is not appropriate. Instead, we are more interested in a smooth measure such as the squared difference between prediction and label (*cf.* ibid., pp. 25 sq.).

Clearly, the correct choice of measure depends on the specific learning problem. We require a definition of risk that is general enough to allow for such a choice, while still enabling claims about learnability. To this end, define a new function (*cf.* Shalev-Shwartz and Ben-David 2014, p. 26):

Definition 2.11 (loss function)**.** For any hypothesis class $\mathcal{H}$, domain set $\mathcal{X}$ and label set $\mathcal{Y}$ a *loss function* is a function $\mathcal{L} : \mathcal{H} \times \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}^+$.

We can then integrate this into the risk (*cf.* ibid., p. 26):

$$L_{\mathcal{D}}(h) = \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\mathcal{L}(h, x, y)\right]. \tag{2.3}$$

The empirical risk can be generalized similarly (*cf.* ibid., p. 27):

$$L_S(h) = \frac{1}{m}\sum_{i=1}^{m}\left[\mathcal{L}(h, x_i, y_i)\right]. \tag{2.4}$$

2.1.1.8. Agnostic PAC Learnability

To extend our previous definition of PAC learnability, we introduce a new formulation that takes into account all the relaxations and generalizations specified in the previous section (*cf.* ibid., p. 25):

Definition 2.12 (agnostic PAC learnable)**.** A given hypothesis class $\mathcal{H}$ with an associated loss function $\mathcal{L} : \mathcal{H} \times \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}^+$ is *agnostic PAC learnable*, if a function $m_{\mathcal{H}} : (0,1)^2 \rightarrow \mathbb{N}$ and a learner $\mathcal{A}$ exist, such that for every:

- $\epsilon \in (0,1)$ (*accuracy*) and $\delta \in (0,1)$ (*confidence*),
- environment distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$,

if the *i.i.d.* assumption on the training data S holds, then evaluating $\mathcal{A}(S)$ with $|S| = m \geq m_{\mathcal{H}}(\epsilon, \delta)$ results in a hypothesis $h \in \mathcal{H}$, where with probability of at least $1 - \delta$ (w.r.t. the sampling of S):

$$L_{\mathcal{D}}(h) \leq \min_{h' \in \mathcal{H}} L_{\mathcal{D}}(h') + \epsilon,$$

with $L_{\mathcal{D}}$ as given in eq. (2.3).

For a finite hypothesis class $\mathcal{H}$, we can again give a sample complexity $m_{\mathcal{H}}(\epsilon, \delta)$, such that $\mathcal{H}$ is agnostic PAC learnable with the ERM learner (*cf.* ibid., pp. 31–34):

$$m_{\mathcal{H}}(\epsilon, \delta) \leq \frac{2 \log \frac{2|\mathcal{H}|}{\delta}}{\epsilon^2}.$$

2.1.1.9. Generalized Loss Function

The previous sections all pertain to prediction tasks. In section 2.1.4, we introduce machine learning tasks that go beyond prediction. For these, a loss function $\mathcal{L} : \mathcal{H} \times \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}^+$ might not be applicable. $\mathcal{Y}$ may not even exist. We therefore introduce a more general *example set* $\mathcal{Z}$ with *examples* z, which happens to be $\mathcal{X} \times \mathcal{Y}$

for prediction tasks (*cf.* ibid., p. 26). We can then define the *generalized loss function* $\mathcal{L} : \mathcal{H} \times \mathcal{Z} \rightarrow \mathbb{R}^+$ (*cf.* ibid., 26 sq.). Accordingly, $\mathcal{D}$ describes the environment distribution over $\mathcal{Z}$. The respective modifications to risk and empirical risk follow intuitively. Agnostic PAC learnability still holds in this general formulation.

With this formulation, we can fully describe a machine learning *task* by its example set $\mathcal{Z}$, environment distribution $\mathcal{D}$ and loss function $\mathcal{L}$. In practical applications, samples $z \sim \mathcal{D}$ are given because the exact $\mathcal{D}$ is unknown or intractable.

2.1.1.10. Summary

The tasks, methods and models described in the remainder of this thesis respect the definitions in the previous sections unless specified otherwise. With agnostic PAC learnability for generalized loss functions, we can reasonably expect a certain risk level with some probability, given sufficient training data and correct inductive biases. PAC learnability can be generalized even further, *e.g.*, to infinite hypothesis classes with certain properties (*cf.* ibid., p. 48).

In the following, we discuss implementations of the aforementioned concepts to obtain a working system. For example, a complete machine learning system can be constructed using the hypothesis class detailed in section 2.1.2 combined with an ERM learner. The optimization problem can be solved with one of the methods from section 2.1.3.

2.1.2. Linear Model

Restricting the hypothesis class $\mathcal{H}$ is a way of introducing prior knowledge, and therefore an inductive bias. A simple, but important class of hypotheses are *linear models*:

$$\mathcal{H}_\phi = \{ \boldsymbol{x} \mapsto \phi(\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b), \boldsymbol{w}, \boldsymbol{x} \in \mathbb{R}^d, b \in \mathbb{R} \}, \tag{2.5}$$

where $\phi : \mathbb{R} \rightarrow \mathcal{Y}$ is a function that maps the scalar result of the affine transform $\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b$ to the label set $\mathcal{Y}$ (*cf.* ibid., 89 sq.). For example, a binary classifier could use $\phi(s) = \operatorname{sgn}(s)$ for $\mathcal{Y} = \{-1, 1\}$. Different ϕ result in different hypothesis classes.

2.1.2.1. Linear Separability

Definition 2.13 (linear separability)**.** If a problem defined by an environment distribution $\mathcal{D}$ is realizable (see section 2.1.1.5) in $\mathcal{H}_{\text{sgn}}$, it is linearly separable. The term can also be applied to specific training data.

For linearly separable problems, the ERM learner w.r.t. $\mathcal{H}_{\text{sgn}}$ can also be given as a linear program (*cf.* ibid., p. 91). Furthermore, the hypothesis space can be made equivalent to the one used by the perceptron with the correct ϕ (see section 2.4.1.1). A common example to illustrate the limits of linear separability is the XOR problem (see fig. 2.1).

2.1.2.2. Logistic Regression

Equation (2.5) is also known as a generalized linear model in literature (*cf.* Bishop 2008, p. 180). The choice of ϕ determines its functionality and interpretation. For example, from a probabilistic viewpoint, $\phi = \operatorname{sgn}$ defines a hypothesis such that:

$$h(x) = \operatorname*{arg\,max}_{y \in \{-1,1\}} \mathbb{P}(y|x)\,,$$

where $\mathbb{P}$ is the distribution as estimated by the generalized linear model.

Knowing this distribution, we can make more fine-grained predictions. For example, an application might make use of the probability of a certain class, given a domain point x. We can formulate such a hypothesis:

$$h(x) = \mathbb{P}(y|x)\,,$$

with $\phi(s) = \sigma(s) = \frac{1}{1+\exp(s)}$. This approach is named logistic regression (*cf.* Shalev-Shwartz and Ben-David 2014, 97 sq.; *cf.* Bishop 2008, 205 sq.). It can also be implemented using neural networks and a sigmoid activation function (see sections 2.4.1.3 and 2.4.1.4). Support Vector Machines (SVMs) (*cf.* Bishop 2008, 325 sqq.) are a further possible implementation.

2.1.3. Gradient-Based Optimization

While ERM is a clearly defined optimization problem, it cannot be implemented directly in most cases. Searching the space of all hypotheses $\mathcal{H}$ for the optimal h^* is only an option for very small $\mathcal{H}$. An exhaustive search is impossible if $\mathcal{H}$ is infinite, *e.g.* linear models (see section 2.1.2). However, we can adjust the problem in many helpful ways.

First, we observe a property of the linear model's hypothesis class. It is *parameterized* by $\boldsymbol{w} \in \mathbb{R}^d$ and $b \in \mathbb{R}$. Instead of searching the hypothesis class, we can express the ERM learner (see eq. (2.1)) as an optimization over $\mathbb{R}^d \times \mathbb{R}$:

$$\begin{aligned} \operatorname{ERM}(S) &= \operatorname*{arg\,min}_{h} L_S(h) \\ &= h(\cdot;\boldsymbol{w}^*, b^*), \text{where} \\ \boldsymbol{w}^*, b^* &= \operatorname*{arg\,min}_{\boldsymbol{w},b} L_S(h(\cdot;\boldsymbol{w}, b))\,. \end{aligned} \tag{2.6}$$

Our search space now has clearly defined structure: a real vector space. We denote this parameter space Θ and a specific instance $\boldsymbol{\theta}$. The optimal hypothesis is described by $\boldsymbol{\theta}^*$.

Second, we allow local minima. We accept that computing the globally optimal hypothesis is infeasible and settle for one that is optimal within a small neighborhood in Θ (*cf.* Goodfellow, Bengio, and Courville 2016, 82 sqq.).

Finally, L, and by extension $\mathcal{L}$ and h are assumed to be differentiable functions w.r.t. $\boldsymbol{\theta}$. If $\boldsymbol{\theta}^*$ is a local optimum, then:

$$\frac{\partial}{\partial \boldsymbol{\theta}} L_S(h(\cdot;\boldsymbol{\theta}^*)) = \mathbf{0}\,.$$

With these assumptions, we can apply the *gradient descent* algorithm (Cauchy 1847). Starting with a random initial value $\boldsymbol{\theta}^{(0)}$ (see also section 2.4.3.1), we use the following update rule:

$$\boldsymbol{\theta}^{(k)} = \boldsymbol{\theta}^{(k-1)} - \eta \frac{\partial}{\partial \boldsymbol{\theta}} L_S(h(\cdot;\boldsymbol{\theta}^{(k-1)}))\,, \tag{2.7}$$

where η is the *learning rate*. The learning rate is not part of Θ and is not optimized using the training data. Such parameters are called *hyperparameters* (*cf.* Goodfellow, Bengio, and Courville 2016, p. 98) (see section 2.2.2). It is common practice to change the learning rate as optimization progresses. Such methods are called learning rate schedules and are explained in more detail in section 6.2.2.3.

2.1.3.1. Stochastic Gradient Descent

The update rule eq. (2.7) minimizes the loss function over the whole training data in each step. While this approach is a correct to the optimization problem, it has two potential disadvantages. First, large datasets such as ImageNet-1k (see section 6.2.1.1) easily overwhelm the working memory of current computers and graphics cards, making the approach infeasible. Second, while local minima are considered acceptable if the respective neighborhood is sufficiently large, minima that are "too local" are still undesirable. However, the update rule eq. (2.7) will get "stuck" in any local minimum regardless of its spatial extent.

Stochastic gradient descent (SGD) is one solution to these problems (*cf.* ibid., 151 sq.). For each iteration k of the update rule, instead of the whole set S, we randomly select a *minibatch* $S_k \subset S$ and minimize L_{S_k} instead of L_S. Minibatches are small enough for efficient computation, with a typical setting of 32 (*cf.* Masters and Luschi 2018). Because they are different for each execution of the update rule, there is a high probability of escaping very local minima of the whole training data simply because they do not exist in all possible minibatches.

2.1.3.2. Momentum

SGD is often combined with a second strategy for avoiding local minima which is called *momentum* (*cf.* Goodfellow, Bengio, and Courville 2016, 296 sq.). Instead of computing a direction and distance in Θ for each step k, we estimate a *change* in direction from the previous step. Effectively, the movement through Θ has a velocity or a momentum. A hyperparameter α is introduced to control the influence of the previous step on the current step. The momentum update rule is defined as:

$$\boldsymbol{\theta}^{(k)} = \underbrace{\alpha(\boldsymbol{\theta}^{(k-1)} - \boldsymbol{\theta}^{(k-2)})}_{\text{momentum}} + \boldsymbol{\theta}^{(k-1)} - \eta \frac{\partial}{\partial \boldsymbol{\theta}} L_S(h(x; \boldsymbol{\theta}^{(k-1)}))\,.$$

A common choice for α is 0.90. There are alternative formulations which reuse gradients from previous steps instead of the positions in Θ. This is done to reduce the exponential influence of α.

2.1.3.3. Adam

The idea of gradient descent with momentum (section 2.1.3.2) can be generalized to estimating statistical moments of individual components of the gradient, which are then used to influence the movement through Θ. Adam, which is proposed in Kingma and Ba (2014), is an implementation of such a method using first- and second-order gradient statistics to calculate a separate learning rate for each dimension of $\boldsymbol{\theta}$. It has empirical advantages over SGD with momentum in many cases. Although it has a larger number of hyperparameters, they are less sensitive to change and present a larger operating range in practice. As such, Adam is a reasonable choice when there is little time for hyperparameter optimization.

2.1.4. Supervised Learning and Alternatives

In the previous section, our overall goal is making predictions. Given an input $x \in \mathcal{X}$, we predict the corresponding label $y \in \mathcal{Y}$ by using our hypothesis. A problem of this sort is called a *supervised learning* problem, if there exists labeled training data $S = \big((x_1, y_1), \ldots, (x_m, y_m)\big)$. In the following, we explore alternatives to this formulation. Further variants are discussed in section 4.2.4 and there is an effort to formalize different levels of supervision proposed in Damen and Wray (2020).

2.1.4.1. Unsupervised Learning

Machine learning problems do not always require predictions to be made. There are numerous other tasks (*cf*. Goodfellow, Bengio, and Courville 2016, 99 sqq.) or problem formulations (*cf*. ibid., 104 sqq.) that involve machine learning. *Unsupervised learning* removes the label set from the learning problem entirely. Intuitively, it would seem that there no longer is a problem. Certainly, the risk is now ill-defined. However, several interesting tasks can be formulated using only the domain set.

For example, exploring a new dataset and finding noteworthy examples, or clusters, without additional information. Another task that enjoys recent popularity is the generation of new examples, specifically images. Generative Adversarial Networks (GANs), presented in Goodfellow et al. (2014), are groundbreaking method in this field. Learning a compact representation of an unlabeled set of images also has potential applications in compression. Section 4.2.4.1 gives an overview of relevant literature relating to unsupervised learning.

2.1.4.2. Semi-Supervised Learning

We can also imagine a middle ground between supervised learning and unsupervised learning. Such a *semi-supervised learning* problem starts with a basic supervised learning task. On top of the labeled training data $S = \big((x_1, y_1), \ldots, (x_m, y_m)\big)$, we add unlabeled training data $S' = (x_{m+1}, \ldots, x_{m+m'})$ (*cf*. Goodfellow, Bengio, and Courville 2016, 243 sq.).

While S' would not suffice on its own to approximate a target function, it still contains useful information. For example, it could be used to learn a better feature representation (see *e.g.*, sections 2.3.4.1 and 2.4). We discuss implementations in section 4.2.4.2. Learning from imprecise data (see section 5.1) is a generalization of semi-supervised learning.

2.1.4.3. Self-Supervised Learning

Although it is not a problem formulation in and of itself, *self-supervised learning* is important to mention here. It is a class of methods to tackle semi-supervised learning problems where the unlabeled training data is labeled using the hypothesis itself. Then, the newly labeled data is fed back into the learner to retrieve an improved hypothesis. Self-supervised learning methods need to be tuned carefully to avoid a feedback loop where mispredictions are learned and thus amplified. However, they can achieve remarkable results (*cf.* Geirhos et al. 2020). Zhai et al. (2019) is a notable example of this technique, with further works mentioned in section 4.2.4.2. We propose a self-supervised method in section 5.3.3.

Such techniques can be applied to semi-supervised learning tasks, but also to *weakly supervised learning*. This paradigm considers training data where some or all of the labels are qualitatively worse, or “weak”. Learning from imprecise data, which is discussed in section 5.1, is such a task.

2.2. Model Selection and Complexity

In section 2.1.1.3, we discuss the notion of risk, which is also called “true error”(see Shalev-Shwartz and Ben-David 2014, p. 14). The overall goal of machine learning is risk minimization, however this cannot be tackled directly unless the environment distribution $\mathcal{D}$ is known exactly. Instead, we solve the proxy problem of ERM. The main issue with ERM is that its optimum can differ significantly from the “true” minimal risk hypothesis. There are only weak bounds that relate the two (see section 2.1.1.8).

For example, consider the *nearest neighbor classifier*, which uses the label of the training data element closest to the domain point in question as a prediction. Unless there are ambiguous samples in the training data, the empirical risk of this classifier is always zero. This does not imply that it generalizes well, *i.e.*, that the true risk is zero. If it does not, the classifier exhibits *overfitting* (*cf.* ibid., 15 sq.). It fits the training data better than the actual environment distribution.

Non-representative training data, or too little training data considering the sample complexity of the given hypothesis class $\mathcal{H}$ can contribute to overfitting. It can also be caused by selecting the wrong model or assuming false inductive biases (see section 2.1.1.5). The opposite phenomenon, *underfitting*, is also possible when the hypothesis class is not complex enough.

Formally, the true error of a hypothesis h_S obtained from an ERM learner $\mathcal{A}_{\text{ERM}}(S)$ can be separated into two types of error: (*cf.* ibid., 40 sq.)

$$L_{\mathcal{D}}(h_S) = \epsilon_{\text{app}} + \epsilon_{\text{est}}\,,$$

where:

- $\epsilon_{\text{app}} = \min_{h' \in \mathcal{H}} L_{\mathcal{D}}(h')$, the *approximation error*, is the lowest possible true error given $\mathcal{H}$ and $\mathcal{D}$. It is determined by the inductive biases and choice of $\mathcal{H}$. If the inductive biases are too strong, or $\mathcal{H}$ is not complex enough, a high approximation error occurs, and vice versa. Note that $\epsilon_{\text{app}} = 0$ iff $\mathcal{D}$ is realizable in $\mathcal{H}$.

- $\epsilon_{\text{est}} = L_{\mathcal{D}}(h_S) - \epsilon_{\text{app}}$, the *estimation error*, is the difference between the approximation error ϵ_{app} and true error achieved by $\mathcal{A}_{\text{ERM}}(S)$. In other words, it represents the error caused by minimizing the empirical risk using training data instead of the true risk using the environment distribution. Consequently, ϵ_{est} can be reduced by a larger training data set. However, it increases with the complexity of $\mathcal{H}$ as opposed to the approximation error, which decreases.

The opposite effects of the complexity of $\mathcal{H}$ on approximation and estimation error are also termed *bias-variance trade-off*. With knowledge of the individual contributions towards the true risk $L_{\mathcal{D}}$, we discuss a further aspect of machine learning. *Model selection* is the meta-problem of adjusting the complexity and selecting the optimal hypothesis class to minimize the true risk.

In the following, we discuss how to control under- and overfitting by adjusting the complexity (section 2.2.1). Furthermore, we explore methods of actually estimating the true error, measuring generalization and overfitting (section 2.2.2).

Complexity in the context of statistical learning theory is often defined as $|\mathcal{H}|$, the cardinality of the hypothesis class. (Agnostic) PAC learnability (see section 2.1.1.8) assumes a finite $\mathcal{H}$ as well. However, most hypothesis classes used in practice are infinite (see sections 2.1.2 and 2.4.1.1). Their complexity can be measured in terms of *VC-dimension* (Vapnik and Chervonenkis 1971) or *Rademacher complexity* (Shalev-Shwartz and Ben-David 2014, 325 sqq.).

2.2.1. Regularization

In this section, we examine a learning paradigm that allows for fine-grained control over the complexity of hypothesis classes. Consider the parameterized version of ERM as described in eq. (2.6), with which we can apply gradient-based optimization methods. A simplified variant of the problem, where we consider only one parameter $\boldsymbol{\theta}$ is defined as follows:

$$\min_{\boldsymbol{\theta}} L_S(h(x;\boldsymbol{\theta}))\,.$$

We now include a *regularization* term $R(\boldsymbol{\theta})$ with $R : \mathbb{R}^d \to \mathbb{R}$ to formulate the regularized loss minimization (RLM) problem (*cf.* ibid., p. 137):

$$\min_{\boldsymbol{\theta}} L_S(h(x;\boldsymbol{\theta})) + R(\boldsymbol{\theta})\,,$$

where $R(\boldsymbol{\theta})$ is a measure of the complexity of the hypothesis class containing the hypothesis associated with $\boldsymbol{w}$.

Every value of $R(\boldsymbol{\theta})$ corresponds to a specific subset of $\mathcal{H}$. With this construction, the bias-variance trade-off is an explicit part of the optimization problem. The choice or parameterization of $R(\boldsymbol{\theta})$ remains part of the model selection problem.

2.2.1.1. Tikhonov Regularization

A very simple form of regularization is *Tikhonov regularization*. It is based on the assumption that the norm of $\boldsymbol{\theta}$ is an indicator of the complexity, or irregularity of the matching hypothesis class. For example, penalizing the norm of weights of a linear model (see section 2.1.2) would limit the slope of the decision function, and in turn reduce the model's susceptibility to noise. However, it also introduces a bias, resulting in a high approximation error, should the task actually have too strong a slope.

One example of a Tikhonov regularizer is the L2 regularizer (*cf.* ibid., p. 138):

$$R(\boldsymbol{\theta}; \beta) = \beta \|\boldsymbol{\theta}\|^2 ,$$

where β is a coefficient, or hyperparameter, to control the amount of regularization. With this parameterization, the model selection meta-problem reduces to an optimization problem over the continuous β. The Tikhonov regularizer can be generalized to any norm, not only the 2-norm.

2.2.1.2. Early Stopping

When using gradient-based optimization methods (see section 2.1.3), the number of optimization steps is an important hyperparameter. It is typically set based on convergence criteria to obtain the lowest possible loss (*cf.* Goodfellow, Bengio, and Courville 2016, p. 643).

However, the number of iterations also affects the complexity of the hypothesis. When the true risk is measured during optimization, it first decreases as the model fits the data better and better, and then increases again as a result of overfitting. *Early stopping* sacrifices convergence criteria based on (empirical) loss in favor of a lower true error (*cf.* ibid., 246 sq.). Effectively, the number of optimization steps is another hyperparameter to be determined during model selection.

Note that early stopping is considered a regularization method in terms of managing a hypothesis' complexity, but is not an instance of RLM.

2.2.2. Validation

In the previous section, we discuss methods of controlling the complexity of hypotheses. Using these methods, we can obtain a parameterized variant of the model selection problem. With parameters $\boldsymbol{\theta} \in \Theta$ of a parameterized hypothesis h, hyperparameters $\omega \in \Omega$ and an environment distribution $\mathcal{D}$ with training data S, we can formalize the model selection problem as:

$$\min_{\omega \in \Omega} L_{\mathcal{D}}(h(\cdot; \boldsymbol{\theta}^*(\omega))) ,$$

where

$$\boldsymbol{\theta}^*(\omega) = \arg\min_{\boldsymbol{\theta}} L_S(h(\cdot; \boldsymbol{\theta})) + R(\boldsymbol{\theta}; \omega) .$$

This formulation does not directly solve the problems of ERM, because it requires the impossible evaluation of $L_{\mathcal{D}}$. Still, it is possible to estimate the true risk and select appropriate hyperparameters ω. In the following discuss various methods of estimating the true risk using separate data sets. They are considered instances of *validation* (*cf.* Bishop 2008, p. 32).

2.2.2.1. Estimating True Risk: Training, Validation and Test Sets

One straightforward way of approximating the true risk is using a *validation set* (*cf.* Bishop 2008, 32 sqq.). Such a set T consists of samples from the environment distribution $\mathcal{D}$. It should be sampled in the same way as the training data it, independently and identically distributed. However, its use should be limited to estimating the true risk to perform model selection. The samples should never be used for learning, *i.e.*, minimizing the empirical risk.

There is one philosophical issue with validation. Model selection is a learning problem as well. While the validation set is used to find the optimal hyperparameters ω and the training data for the parameters $\boldsymbol{\theta}$, respectively, the distinction is rather arbitrary. Thus, it is possible to overfit the model selection problem on a given combination of validation set and training data. For example, consider a random seed that is used to sample the initial parameters $\boldsymbol{\theta}^{(0)}$ for gradient descent. The seed itself would be thought of as a hyperparameter, but it can be used, with enough time, to "learn" the validation set.

A solution to this issue is a *held-out test set* also sampled *i.i.d.* from $\mathcal{D}$. This held-out test should be used only once, to validate the results of the model selection process.

> While ideally the validation and test sets are additional samples from the distribution $\mathcal{D}$, it is common practice to split existing training data into training, validation and test parts. If the *i.i.d.* assumption holds, there is no difference except for the sample size.

2.2.2.2. Cross-Validation

The split into training, validation and test sets discussed in the previous section is typically determined only once and not changed subsequently. However, which examples end up in which split can significantly affect the results of the model selection process if the dataset is small to begin with.

There are alternatives for such situations that rely on combining different ways of splitting a small dataset. *k-fold cross-validation* (*cf.* ibid., p. 33) is such a method. The dataset is first split (evenly) into k subsets. Then, one of the subsets is used as a validation set, while the rest constitutes the training data. Each of the k different combinations is evaluated independently, resulting in a better approximation of the risk. Cross-validation effectively controls for the effects of splitting a dataset in different ways.

For even smaller datasets, *leave-one-out cross-validation* is an option. Instead of splitting into k subsets, where usually $k << |S|$, each example is considered a single subset. This results in a very large number of possible combinations.

2.2.2.3. Practical Considerations

We generally assume that training data is sampled independently and identically (*i.i.d.*) from $\mathcal{D}$. However, there are cases in practice where this assumption does not hold. Certain examples in the training data may be correlated, which needs to be considered when selecting a subset for validation. If validation and training examples correlate, then validation no longer approximates the true risk.

For example, consider a medical dataset consisting of many slices of computed tomography (CT) scans. If there are 1248 slices captured from 63 patients, the slices cannot be assumed fully independent. As a precaution, when splitting the dataset, slices from the same patient should not be in multiple splits. It is a common practice to adapt leave-one-out cross-validation on a patient level, *i.e.*, *leave-one-patient-out* (*cf.* Häfner et al. 2012).

Time series, *e.g.*, videos, financial data, or climate data, pose another problem. All samples are correlated by design, as a result of natural laws. When splitting such datasets, the sequence needs to be considered. For example, videos should not be split on a frame-by-frame basis, but into cohesive segments.

2.3. Machine Learning for Computer Vision

The previous sections (sections 2.1 and 2.2) focus on very general machine learning concepts and methods. Specifically, we pose no requirements towards the domain set $\mathcal{X}$, except in section 2.1.2, where we assume a real vector space. In this section, we explore the area of *computer vision*, where methods of machine learning are applied to solve a variety of tasks specific to visual information.

We begin with a formal definition of images, which are the most common data type in computer vision, and their digital representation in computers. Afterwards, a selection of relevant tasks is discussed. Finally, we address the special "needs" of machine learning methods when processing images, and methods to combat the challenges involved in machine learning for computer vision.

2.3.1. Images

When we mention images in this thesis, we always refer to a representation that is suitable for processing by a (digital) computer. However, for the purpose of understanding the image formation and acquisition process, we start with a continuous definition (*cf.* Gonzalez and Woods 2018, p. 18):

Definition 2.14 (Image function)**.** An *image function* with c color channels is a two-dimensional function $f : \mathbb{R}^2 \to \mathbb{R}^c$ with $c \in \mathbb{N}$.

While planar image functions are sufficient for the applications discussed in this thesis, the definition could be adapted to other situations, *e.g.*, moving images, or volumes.

2.3.1.1. Discretization

Both the domain and the range of an image function are continuous, and thus unsuitable for digital storage and processing. They need to be discretized in order to obtain a digital image, which is defined as follows:

Definition 2.15 (Image)**.** A digital *image* with c color channels is a three-dimensional array $\boldsymbol{F} \in \mathcal{C}^{w \times h \times c}$, where w and h are width and height, respectively. $\mathcal{C}$ denotes the *color space* of the image, which is discussed in section 2.3.1.2.

An individual element of $\boldsymbol{F}$ is called a *pixel* (short for picture element, *cf.* Gonzalez and Woods 2018, p. 18). The discretization first involves the domain of the image function — the coordinates. This step is called *sampling* (*cf.* ibid., 63 sqq.) and is usually part of the acquisition process. For example, a camera sensor is constructed of a fixed number of individual elements which measure light intensity, sampling the image plane by design. If an image function contains high-frequency components, selecting an insufficient amount of pixels for sampling (resulting in a low *sampling frequency*) can lead to destructive artifacts. This phenomenon is called *aliasing* (*cf.* ibid., p. 221).

Obtaining a digital representation of the image function's range is termed *quantization*. While the number of color channels is already discrete, the intensity measurement is not. 8-bit unsigned integers ($\mathbb{F}^{256}$) are a common representation of intensity values (*cf.* ibid., p. 70), and we use it unless stated otherwise. However, a number of machine learning methods are modeled using continuous values, *e.g.*, linear models (section 2.1.2) or CNNs (see section 2.4). In this case, a floating-point representation of intensities is used, where the precision is implementation-dependent.

2.3.1.2. Color Space

As introduced in sections 2.3.1 and 2.3.1.1, images have *channels*, which refer to the dimensionality of the underlying color space. A color space is used to represent both the color and the intensity of light captured. The RGB (red, green, blue) color space is the most common representation in images intended for human viewing on monitors (*cf.* ibid., p. 405). It is also used by most capture devices, *i.e.*, video and photo cameras and scanners (*cf.* ibid., p. 406). If added together, red, green, and blue can be mixed into most colors perceptible by humans, but not the complete visible spectrum of light wavelengths (*cf.* ibid., p. 402).

RGB is represented by a cube $[0, 1]^3$ in image functions or $\mathbb{F}^{256 \times 3}$ in digital images (*cf.* ibid., p. 407), such that a single color pixel contains 24 bits of information. Unless stated otherwise, the RGB color space is assumed throughout this thesis. Alternative additive color spaces exist, *e.g.*, HSI and CIE (*cf.* ibid., pp. 411, 419), which also require three degrees of freedom and can be transformed into one another. For subtractive color mixing, *e.g.*, by printing, the CMYK color space is common (*cf.* ibid., 408 sq.).

2.3.2. Computer Vision Tasks

This section formally introduces the tasks that are the building blocks of many modern computer vision applications based on machine learning. This qualification is important because there are many computer vision methods and tasks that are not associated with machine learning at all. The following tasks are all supervised learning problems (see section 2.1.4) where the domain set is a set of images. They only differ w.r.t. the label set $\mathcal{Y}$.

2.3.2.1. Image Classification

If the label set $\mathcal{Y}$ is a finite set $\mathcal{Y}^P$ with mutually exclusive semantics, *e.g.*, $\mathcal{Y}^P = \{\texttt{cat}, \texttt{dog}, \texttt{tv}\}$, then the prediction task characterized by $\mathcal{D}$ over $\mathbb{I} \times \mathcal{Y}$ is called an *image classification* task (*cf.* Gonzalez and Woods 2018, p. 43; *cf.* Gonzalez and Woods 2018, 903 sqq.). Image classification is by far the most common computer vision task.

There are several popular datasets and benchmarks available, which we explore in detail in section 6.2.1.1. This thesis focuses on classification tasks as well. Moreover, a number of datasets used in this thesis represent *fine-grained recognition* tasks, *e.g.*, differentiating between very similar species of birds. However, the exact definition of fine-grained is subject of debates (*cf.* Duan et al. 2012; Chang et al. 2021). Further work in fine-grained recognition can be found in section 4.2.2.

2.3.2.2. Object Detection

Image classification is limited by a shallow description of the image's content in terms of a single label. If we aim for a deeper understanding of an image, or a richer description of its contents, *object detection* is the logical next step. This task combines two subtasks (*cf.* Goodfellow, Bengio, and Courville 2016, p. 453).

First, possible objects in an image are localized. We describe the region these *instances* occupy with an axis-aligned *bounding box*. For example, in a two-dimensional image, such a bounding box is identified in $\mathcal{Y}^{BB} = \mathbb{R}^4 = \mathbb{R}^2 \times \mathbb{R}^2$ using the coordinates of the top-left and bottom-right vertices. Second, each localized object is classified in a label set $\mathcal{Y}^P$, assuming the same semantics as for image classification.

The label set of the object detection task is a power set because the number of objects in any given image is not fixed, such that $\mathcal{Y} = \mathcal{P}(\mathcal{Y}^{BB} \times \mathcal{Y}^P)$. This representation poses challenges for certain models with fixed dimensionality (for example YOLO, see section 7.2.1.2).

2.3.2.3. Semantic Segmentation

We can further generalize object detection w.r.t. its spatial component. *Semantic segmentation* (*cf.* Gonzalez and Woods 2018, 699 sqq.) is a task where each individual *pixel* of an image is classified in a given $\mathcal{Y}^P$. The label set is a set of functions $\mathcal{Y} = \{f : \{0, \dots, w\} \times \{0, \dots, h\} \rightarrow \mathcal{Y}^P\}$.

Since labels are essentially images themselves, the hypothesis space is even larger than for image classification or detection. The consequences are discussed in section 2.3.3.1. However, semantic segmentation can be simplified by transforming the problem into many individual classification problems of local neighborhoods (*e.g.* Brust et al. 2015a). This is a trade-off because smaller neighborhoods result in less *context* information to help the classifier.

2.3.3. Challenges

Computer vision faces various challenges, most of which are inherited from machine learning in general. There are also unique challenges resulting from the nature and representation of images. We address the effects of the high dimensionality of images compared to other types of data and their particular technical requirements in the following.

2.3.3.1. Curse of Dimensionality

The "curse of dimensionality", introduced in Bellman (1957), describes the exponential increase in volume when dimensions are added to a vector space. As a result, it becomes increasingly hard to sample the space with a certain density. However, representative samples as training data are a fundamental requirement of machine learning. The UCI repository[1], a common benchmark dataset for general machine learning, contains examples with tens of dimensions. In contrast, CIFAR-100 (section 6.2.1.1), a dataset of comparatively small images, already has 3072 dimensions. Larger images have hundreds of thousands of dimensions. Intuitive reasoning around high-dimensional data is hard for humans as the relation between distance and volume behaves unexpectedly (*cf.* Bishop 2008, p. 36).

At the same time, the number of data points is not very high, ranging from tens of thousands to millions at most. Consequently, this training data would not appear as representative as required by theory. Still, real-world applications of computer vision are feasible and practical. It is assumed that images specifically only occupy a small, lower-dimensional subspace of their respective space $\mathcal{C}^{w \times h \times c}$ — the *natural image manifold* (*cf.* ibid., 37 sq.). The idea of invariances is closely related to this assumption and discussed in section 2.3.4.2.

2.3.3.2. Computational and Storage Constraints

Recent large-scale benchmark datasets are many, up to hundreds, of gibibytes in size when compressed, *e.g.*, OpenImages-v6[2] and ImageNet-1k (see section 6.2.1.1). There exist datasets that are too large to be stored on single computers, *e.g.*, the JFT-300M dataset described in Sun et al. (2017a), or an internal dataset used by Facebook (Mahajan et al. 2008) comprised of billions of images.

Combined with the curse of dimensionality discussed in the previous section, image are a demanding modality compared to, *e.g.*, sound or financial time series. Not only does a single example require kibibytes or mebibytes to store, the number of data points needs to be higher than other modalities as well. Furthermore, randomly sampled data contains redundant elements (*cf.* Birodkar, Mobahi, and Bengio 2019).

Processing is a further challenge. Contemporary methods cannot be used on commodity compute hardware. Instead, they require massively parallel processing units such GPUs or even task-specific integrated circuits (*cf.* Wang, Wei, and Brooks 2019). This widespread adoption leads to excessive energy use and in turn, calls for policies around the use of such methods (*cf.* Strubell, Ganesh, and McCallum 2019). If compute resources are constrained, which occurs in mobile devices or edge computing scenarios, special solutions are available, *e.g.*, as discussed in Wang et al. (2020). However, these methods trade off accuracy in favor of runtime or memory requirements. In contrast, tasks such as speech recognition can be solved in real-time with limited resources (*cf.* Jo et al. 2019).

---

[1] `https://archive.ics.uci.edu/ml/index.php` (last accessed April 14th, 2021).

[2] `https://storage.googleapis.com/openimages/web/factsfigures.html`

2.3.4. Feature Extraction and Invariances

All the challenges mentioned in the previous section may lead one to believe that solving the computer vision tasks described in section 2.3.2 using machine learning methods is exceedingly hard or impossible. However, that is only the case when considering machine learning methods that are not image-specific, *e.g.* SVMs, but applied directly to images. In the following, we discuss alternative representations of images that are more suitable for these methods. We also discuss several invariances that can be exploited in computer vision tasks.

2.3.4.1. Feature Representations

Images represented digitally as matrices are typically not suitable for general machine learning methods. Instead, we select different representations called *features*. Features are "attributes that [...] are going to be of value in differentiating between entire images or families of images" (Gonzalez and Woods 2018, p. 812).

Ideally, a feature representation has fewer dimensions than the respective image (see section 2.3.3.1). It should also be constructed to fit the chosen machine learning method. For example, features should be linearly separable if a linear model is used (see section 2.1.2). Features can be "hand-crafted", *e.g.*, by considering several invariances of images, as discussed in the following section. Building bespoke feature representations for specific tasks is common. The spatial ray features described in Kühnl, Kummert, and Fritsch (2012) are a good example of features that are only really suitable for their intended task. There are also more generally applicable features, *e.g.*, histogram of oriented gradients (HoG) introduced in Dalal and Triggs (2005).

Feature representations can also be learned from data given sufficient quantities. Such methods are commonly considered deep learning methods (see section 2.4), if the features are learned *end-to-end*, *i.e.*, together with the classifier or regressor.

2.3.4.2. Invariances

As described in section 2.3.3.1, the ratio between the number of examples in the training data and the number of dimensions of the domain set is important for the success of any machine learning application. Images are especially problematic because of their very high dimensionality. Part of the "value" of features is that they can reduce dimensionality substantially without loss of relevant information. This reduction is achieved through *invariances*.

A task is said to be invariant to a certain transformation on the domain set, if the transformation does not change the label associated with the domain point. It is important to note that invariances differ strongly between tasks. The following invariances are frequently exploited to build features for general object recognition tasks (*cf*. Gonzalez and Woods 2018, p. 812):

- *Translation* When an image is shifted slightly in any direction, the label in terms of classification should not be affected. In the case of object detection and semantic segmentation, it changes predictably through the same translation as applied to the image. This change is called *covariant* (*cf*. Gonzalez and Woods 2018, p. 812), and is fundamental property of convolutional layers (see section 2.4.2.2).

- *Rotation* Similar to translation, this geometric transformation is not expected to change the classification of an image, and applies to bounding boxes and segmentation maps in the same way.
- *Scale* The description of an object does not change depending on the distance to the camera. Similarly, scale should not affect classification, and transform any label in the same way as translation and rotation.

All of the transformations listed above are coordinate transforms on the images. Translation and rotation are rigid transformations as they preserve the (euclidean) distance between coordinates. Scale is an affine transformation. In all three cases, the invariance is not global: it is limited to a certain extent, *e.g.*, because important objects may move outside the image.

In addition, there is a further invariance that is not geometric in nature:

- *Illumination* The semantics, the position and the boundary of objects should not be dependent on the lighting, as long as the visibility is not affected. Any change of lighting, over position in the image, or over time (in the case of videos) should not have an effect.

2.4. Deep Learning

This section introduces a set of methods commonly known as *deep learning*. One important ingredient is the learning of a feature representation from large amounts of data, replacing “hand-crafted” features (Goodfellow, Bengio, and Courville 2016, p. 4). *Representation learning* is made possible by very large (and deep!) *neural networks*, which are also associated with the term deep learning. They are called convolutional neural networks (CNNs). The remainder of this section describes these networks in detail, proceeding in a roughly chronological order.

2.4.1. Artificial Neural Networks

CNNs are a special variant of *artificial* neural networks suitable for large-scale image and signal processing and capable of representation learning. We first discuss neural networks in general to build a theoretical foundation.

The first mention of neural networks is found in McCulloch and Pitts (1943). Inspired by the human brain and nervous system, McCulloch and Pitts develop a temporal first-order predicate calculus. It is built on a set of axioms about neural interaction derived from theoretical neurophysiology. A neural network consist of neurons and synapses which interconnect neurons. The topology never changes. A special case of these neural networks, namely cycle-free topologies, are comparable to modern neural networks. However, their use of binary logic and some of their other axioms severely limit possible applications.

2.4.1.1. Perceptron

The *perceptron* (Rosenblatt 1958) is a continuous generalization of the McCulloch-Pitts model. It describes a single neuron in terms of a special case of the linear model (see section 2.1.2). The perceptron hypothesis is given by (*cf.* Bishop 2008, p. 192):

$$h(\boldsymbol{x}; \boldsymbol{w}, b) = \phi(\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b) \,, \tag{2.8}$$

with $\boldsymbol{w}, \boldsymbol{x} \in \mathbb{R}^d, b \in \mathbb{R}$ and $\phi(s) = \mathbb{1}[s \geq 0]$. In the context of neural networks we call ϕ an *activation function*. $\boldsymbol{x}$ represents the input to the neuron in terms of a d-dimensional vector. The parameters $\boldsymbol{w}$ and b are called *weights* and *bias*, respectively.

2.4.1.2. Multi-Layer Perceptron

The perceptron in eq. (2.8) can realize (w.r.t. the realizability assumption) the same tasks as the linear model (section 2.1.2). The task, or the training data, has to be linearly separable (see section 2.1.2.1). In practice, this is problematic. One can easily imagine a trivial task which is not linearly separable by any means (see fig. 2.1). This limitation affects single perceptrons. However, multiple perceptrons can be arranged into *layers* and combined to solve arbitrarily complex tasks (in terms of function approximation, *cf.* ibid., p. 230). The result is a special neural network: a multi-layer perceptron (MLP). Its hypothesis is defined as:

$$h(\boldsymbol{x}; (\boldsymbol{W}_l, \boldsymbol{b}_l)_{l=1,\ldots,L}) = (h_L \circ h_{L-1} \circ \ldots \circ h_1)(\boldsymbol{x}) \,, \tag{2.9}$$

with the weights $\boldsymbol{W} \in \mathbb{R}^{d_l \times d_{l-1}}$, biases $\boldsymbol{b} \in \mathbb{R}^{d_l}$, and inputs $\boldsymbol{x} \in \mathbb{R}^{d_{l-1}}$.

$$h_l(\boldsymbol{x}; \boldsymbol{W}_l, \boldsymbol{b}_l) = \phi_l(\boldsymbol{W}_l \boldsymbol{x} + \boldsymbol{b}_l) \,. \tag{2.10}$$

Note that $\boldsymbol{W}$ and $\boldsymbol{b}$ are now a matrix and a vector, respectively. The hypothesis h_l, or *output* of the l-th layer represents the activations of several neurons placed next to each other and all connected to the same set of inputs. Such a layer is called a *fully-connected layer*.

We apply the activation function in a point-wise fashion unless specified otherwise. For the very first layer h_1, the input is a domain point. Every following layer has the output of the previous layer as its input. The output of the final layer is the overall hypothesis of the MLP. Because of this directionality, an MLP is a *feed-forward* neural network.

2.4.1.3. Activation Functions

In the linear model, the function ϕ has the purpose of mapping the intermediate result $s = \langle \boldsymbol{w}, \boldsymbol{x} \rangle + b$ to the label set. The activation function of the last layer in an MLP is used in the same manner. However, in eq. (2.10), we can see that all layers have their own activation function, not only the last. These $\phi_1, \ldots, \phi_{L-1}$ are essential, but for a different reason. They serve as non-linearities. Without them, the whole composition $h_{L-1} \circ \ldots \circ h_1$ could be described by a single matrix $\boldsymbol{W}$ and vector $\boldsymbol{b}$, and in turn by a single perceptron (*cf.* Goodfellow, Bengio, and Courville 2016, p. 192). The same is true for linear activation functions.

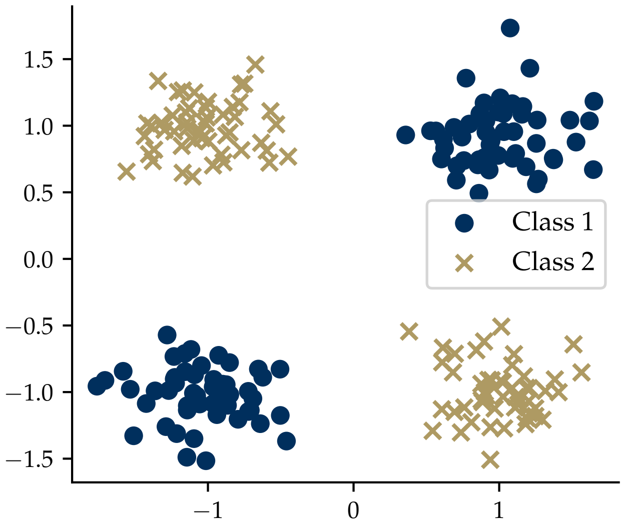


Figure 2.1.: The XOR problem, which is not linearly separable (see section 2.1.2.1).

Activation functions should not only be non-linear, but also differentiable for use with gradient-based optimization techniques (see section 2.1.3). Differentiability rules out the original perceptron activation function $\phi = \operatorname{sgn}$. In the following, we discuss several important activation functions (*cf.* Goodfellow, Bengio, and Courville 2016, 191 sqq.):

- *Sigmoid* $\phi(s) = \sigma(s) = \frac{1}{1+\exp(s)}$. This function is used for logistic regression (see section 2.1.2.2). Its range $[0, 1]$ makes it suitable for approximating individual probabilities.

- *Hyperbolic Tangent* $\phi(s) = \tanh(s) = \frac{\exp(s)-\exp(-s)}{\exp(s)+\exp(-s)}$. Like σ, it has a sigmoidal shape. However, it has a point symmetry around the origin and locally resembles the identity at $\tanh(0) = 0$, which has benefits for the application of neural networks in practice.

- *Rectified Linear Unit "ReLU"* $\phi(s) = \max(0, s)$. First described in Nair and Hinton (2010), this activation function is obtained by taking the limit of many neurons with sigmoidal activation functions and randomly distributed b. It is easy to compute and free of second-order effects because the second derivative is zero almost everywhere (*cf.* Goodfellow, Bengio, and Courville 2016, p. 193). Strictly speaking, it is not differentiable everywhere, which is not problematic for practical purposes.

- *Softmax* $\phi(s)_k = \frac{\exp(s_k)}{\sum_{k'=1}^{d} \exp(s_{k'})}$. This activation function is a common choice for the last layer of a neural network in classification tasks. Its individual components are never negative and always have the sum 1, which makes softmax ideal for categorical probability distributions (*cf.* ibid., p. 81). Furthermore, the combination of softmax activation function and cross-entropy loss function (see section 2.4.1.4) has a derivative that can be calculated very efficiently (*cf.* ibid., p. 222).

2.4.1.4. Loss Functions

Neural networks are learned, or "trained", from data by solving the ERM problem (see section 2.1.1.3). The hypothesis space is parameterized by weights and biases, and the solution is determined using a gradient descent method (see section 2.1.3). For this approach, we require a differentiable loss function. To calculate the derivatives w.r.t. the individual layers' weights, the chain rule is applied. The process is known in literature as *backpropagation* (*cf.* Rumelhart, Hinton, and Williams 1986; Goodfellow, Bengio, and Courville 2016, p. 204).

Depending on the task, the following loss functions are common choices:

- *Mean Squared Error* $\mathcal{L}(h,x,y) = \frac{1}{2}\|h(x) - y\|_2^2$. The mean squared error loss function (*cf.* Goodfellow, Bengio, and Courville 2016, p. 108) is suitable for regression tasks, *i.e.*, prediction of a continuous value. In practice, it is often prefixed with $\frac{1}{2}$ to remove the factor 2 from the derivative. Note that this definition does not contain a mean explicitly. Instead, the mean operator is part of the empirical risk (see eq. (2.4)). Minimizing the squared error is equivalent to maximizing the *likelihood* assuming a normal distribution of y (*cf.* ibid., p. 143).
- *(Categorical) Cross-Entropy* $\mathcal{L}(h,x,y) = -\sum_{k=1}^{d} y_k \log h(x)_k$. When y and h encode categorical probability distributions, this loss function is used for classification and combined with the softmax activation function. It can be derived from likelihood maximization as well (*cf.* ibid., p. 132). There is an alternative formulation for binary classification when the last layer's activation function is a sigmoid, denoted *binary cross-entropy*: $\mathcal{L}(h,x,y) = -y\log h(x) + (1 - y)\log\left(1 - h(x)\right)$.

2.4.2. Convolutional Neural Networks

The MLP in eq. (2.9) is defined for domain sets that are real vector spaces. Images, however, are represented as matrices or objects with three indices (see section 2.3.1). As such, they are incompatible. While images can be "flattened" to vectors, spatial relations in the image cannot be kept intact this way. Furthermore, this loss of information complicates feature learning, since many invariances are defined w.r.t. two-dimensional coordinate transformations.

In this section, we describe a replacement for eq. (2.10) which solves both these problems. It takes the spatial structure of images into account and is translation covariant. This new *convolutional layer* exchanges the matrix-vector product $\boldsymbol{W}_l\boldsymbol{x}$ with a convolution $\boldsymbol{W}_l * \boldsymbol{X}$ (*cf.* ibid., p. 348):

$$h_l(\boldsymbol{X};\boldsymbol{W}_l,\boldsymbol{b}_l)_{i,j,k} = \phi_l\left(\sum_{i',j'}^{w',h'}\sum_{k'}^{d_{l-1}} \boldsymbol{W}_{l;i',j',k',k}\boldsymbol{X}_{i-i'+1,j-j'+1,k'} + \boldsymbol{b}_k\right), \tag{2.11}$$

which transforms an input $\boldsymbol{X}$ of dimensions $w \times h \times c$ into an output of dimensions $(w - w' + 1) \times (h - h' + 1) \times c'$ using a *kernel* $\boldsymbol{W}$ of size $w' \times h' \times c' \times c$. While an image has a number of channels, the output of a convolutional layer has a number of *feature maps*.

2.4.2.1. Implementing Convolutional Layers

Equation (2.11) is a typical implementation of a convolutional layer, but there are many alternatives. This section discusses possible design choices when constructing such a layer.

First, we observe that the central operation in eq. (2.11) is only really a convolution w.r.t. the dimensions of width and height. If we set the kernel's w' and h' sizes to one, we obtain:

$$h_l(\boldsymbol{X}; \boldsymbol{W}_l, \boldsymbol{b}_l)_{i,j,k} = \phi_l \left(\sum_{k'}^{d_{l-1}} \boldsymbol{W}_{l;1,1,k',k} \boldsymbol{X}_{i,j,k'} + \boldsymbol{b}_k \right).$$

Reducing the image dimensions w, h and c to one, and removing all indices of dimensions of size one, the equation becomes:

$$\begin{aligned} h_l(\boldsymbol{X}; \boldsymbol{W}_l, \boldsymbol{b}_l)_k &= \phi_l \left(\sum_{k'}^{d_{l-1}} \boldsymbol{W}_{l;k',k} \boldsymbol{X}_{k'} + \boldsymbol{b}_k \right) \\ &= \phi_l \Big((\boldsymbol{W}_l \boldsymbol{X} + \boldsymbol{b}_k)_k \Big), \text{ or} \\ h_l(\boldsymbol{X}; \boldsymbol{W}_l, \boldsymbol{b}_l) &= \phi_l (\boldsymbol{W}_l \boldsymbol{X} + \boldsymbol{b}), \end{aligned}$$

which is equal to a layer in an MLP. Along the axis of feature maps, the convolutional layer does not in fact convolve image and kernel, but rather computes a matrix-vector product.

The second observation is that the output of a convolutional layer as described in eq. (2.11) has a smaller width and height than the input image. This is necessary in order to keep the indices of $\boldsymbol{X}_{i-i'+1,j-j'+1,k'}$ within the range of the image. This reduction in output size makes the convolution *valid* (*cf.* Goodfellow, Bengio, and Courville 2016, p. 349). Alternatively, one can pad the input image with zeros before convolution such that the input and output width and height of the layer are equal. Padding is not indicated along the channel axis because eq. (2.11) is not a convolution in that respect, as per our first observation.

2.4.2.2. Spatial Pooling

In a convolutional layer, the same filter mask is applied to all spatial locations of the layer's input. This property is known as *weight sharing* (*cf.* LeCun et al. 1988), and means that convolutional layers are translation covariant (see section 2.3.4.2). However, if a neural network should detect the presence of certain objects independent of their location, translation *invariance* would be even better.

Spatial pooling layers can provide limited translation invariance. The layer's input is divided into regions. Commonly, a grid of equally sized non-overlapping regions is used (*cf.* Goodfellow, Bengio, and Courville 2016, p. 342). 2×2 pixels is a typical region size. For each region, an output value representative of all input values inside the region is determined. *Maximum pooling* (*cf.* Zhou and Chellappa 1988) is a common implementation. If the input is translated slightly, such that the maxima remain inside their original regions, the result of maximum pooling is unchanged.

A further benefit of spatial pooling with non-overlapping regions is the dimensionality reduction, which is a possibly greater benefit than the limited translation invariance. However, a similar reduction can be achieved by implementing *strides* inside the convolutional layer (*cf*. Goodfellow, Bengio, and Courville 2016, 348 sq.). This effectively results in a spatial pooling where the representative value is always the upper-left pixel, but at a considerable speed-up compared to maximum pooling.

2.4.3. Implementation Details

While the fundamental building blocks of CNNs are available since the late 1980s (*e.g*. LeCun et al. 1988; Zhou and Chellappa 1988), the first "breakthrough" result (Krizhevsky, Sutskever, and Hinton 2012) of a deep learning system is published in 2012. This coincides with the availability of large-scale labeled training data (see also section 6.2.1.1), and with the advent of general-purpose GPU computing. In fact, the neural network architecture in Krizhevsky, Sutskever, and Hinton (ibid.) is simply the largest that fits the specific GPU hardware and not the result of a model selection process (see section 2.2).

In the following, we discuss a number of smaller "tweaks" and implementation details that are also crucial for the successful application of a CNN.

2.4.3.1. Initialization

We train CNNs using gradient-based optimization (see sections 2.1.3 and 2.4.1.4). Its success is highly dependent upon the correct method of random initialization of the weights and biases. The symmetry of weights is of particular concern. If two weight components in the same layer are initialized to identical values, they remain identical throughout the whole optimization process. This effectively reduces the complexity of the neural network and leads to redundant calculations.

A simple heuristic initialization procedure is offered in LeCun et al. (1988). Given a layer with d_{l-1} input dimensions, each weight component is sampled independently from a uniform distribution $\mathcal{U}$:

$$\boldsymbol{\theta}^{(0)} \sim \mathcal{U}[-\frac{2.4}{d_{l-1}}, \frac{2.4}{d_{l-1}}] .$$

However, this method has a number of drawbacks which are discussed in Glorot and Bengio (2010), including "vanishing gradients". If the activation functions used in the neural network are hyperbolic tangents (see section 2.4.1.3), Glorot and Bengio propose the following initialization for a layer with d_{l-1} input dimensions and d_l output dimensions:

$$\boldsymbol{\theta}^{(0)} \sim \mathcal{U}[-\frac{\sqrt{6}}{\sqrt{d_{l-1}+d_l}}, \frac{\sqrt{6}}{\sqrt{d_{l-1}+d_l}}] .$$

Recent implementations of CNNs often use the ReLU activation function (see section 2.4.1.3) as opposed to the hyperbolic tangent. In He et al. (2015), the authors derive an optimal initialization for such “rectified” neural networks. The weights for a layer with d_{l-1} inputs are sampled from a normal distribution $\mathcal{N}$ as follows:

$$\boldsymbol{\theta}^{(0)} \sim \mathcal{N}(0, \sqrt{\frac{2}{d_{l-1}}}) \,. \tag{2.12}$$

2.4.3.2. Batch Normalization

Gradient descent algorithms (see section 2.1.3) change all weights simultaneously, during each step. However, the change of an individual weight component, as determined by the partial derivative, is only “correct” assuming that the remaining weights are not changed. On the scale of whole layers, this phenomenon is discussed in Ioffe and Szegedy (2015) as “internal covariate shift”.

To counteract this, Ioffe and Szegedy propose a normalization technique called *batch normalization*. The inputs to a layer are normalized element-wise by subtracting the mean and dividing by the standard deviation (*cf*. Goodfellow, Bengio, and Courville 2016, p. 268). During training, the statistics can simply be calculated across a sufficiently large minibatch (see section 2.1.3.1). However, when the model is used for predictions, the minibatch size is effectively 1, which makes the calculation impossible. Instead, moving averages of the normalization coefficients are maintained during optimization and stored for later use in predictions.

2.4.3.3. Residual Networks

In section 2.4.3.1, we already state the importance of initialization. For very deep neural network, *e.g.*, with hundreds of layers, it is even more relevant, as a single badly initialized layer can cause the whole network to fail. If the weights are close zero, information is destroyed.

Residual networks (*cf*. He et al. 2016a,b) are a solution to this problem. Instead of accepting the loss of information with small weights, a reasonable default is provided. For each layer, its input is added to the output, such that zero weights result in an identity layer where no loss of information occurs. Only the deviation from the identity (the residual) has to be learned. A residual convolutional layer is defined as:

$$h_l(\boldsymbol{X}; \boldsymbol{W}_l, \boldsymbol{b}_l)_{i,j,k} = \phi_l\left(\sum_{i',j'}^{w',h'} \sum_{k'}^{d_{l-1}} \boldsymbol{W}_{l;i',j',k',k} \boldsymbol{X}_{i-i'+1,j-j'+1,k'} + \boldsymbol{b}_k\right) + \boldsymbol{X} \,.$$

The handling of edge cases, *e.g.*, $k' \neq k$, is discussed further in He et al. (2016a).

3. Concept Hierarchies as Semantic Knowledge

All I know is that I don't know
All I know is that I don't know nothing
And that's fine. (Operation Ivy)

In this chapter, we abandon the notion of classes in favor of the more general concepts. By machine learning convention, all classes in a set are assumed to be mutually exclusive[1]. Something that is a member of one class cannot be in another class at the same time. Hence, the corresponding random variables are modeled using a categorical distribution.

A concept does not have this limitation. For example, an `English Cocker Spaniel` can be a `dog` at the same time. However, restrictions can still exist on a more complex level. The `English Cocker Spaniel` cannot be a `Pembroke Welsh Corgi` even though both are dogs. This is informed by semantic knowledge.

In the following, these restrictions based on semantic knowledge are modeled by relations between concepts. We then use these relations to nest concepts into hierarchies. Finally, we explore knowledge bases from which such hierarchies can be obtained.

3.1. Formal Introduction

Let us start with a simple prediction task. Examples are from the set $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$, where there exists a domain set $\mathcal{X}$, *e.g.*, images or time series, as well as a label set $\mathcal{Y}$. If we assume that our task is a classification task, then there exists a set of classes $\mathcal{Y}^P$. We then define $\mathcal{Y} = \mathcal{Y}^P$ (see *e.g.*, section 2.3.2.1), such that every domain point x is described by exactly one class. This restriction is reasonable as long as all classes in $\mathcal{Y}^P$ are mutually exclusive. However, consider the set shown in fig. 3.1 as $\mathcal{Y}$. What would be a reasonable label for an image of a `1999 Toyota Camry`? If we label it as `car`, all other possible labels are ruled out. That includes `vehicle` and `object`, which are also reasonable labels. Thus, we need to reformulate $\mathfrak{T}$ to allow for multiple labels at the same time, which we call *concepts* instead of classes to highlight the omission of mutual exclusivity.

A classification problem where $|\mathcal{Y}^P| > 2$, but $\mathcal{Y} = \mathcal{Y}^P$ is called a multi-class classification problem (*cf.* Shalev-Shwartz and Ben-David 2014, 25 sq.; *cf.* Bishop 2008, 182 sqq.). A multi-*label* classification problem generalizes this, such that any number of classes per label is allowed, *i.e.*, $\mathcal{Y} = \mathcal{P}(\mathcal{Y}^P)$. This formulation allows a label like $\{\texttt{object}, \texttt{vehicle}, \texttt{car}\}$. However, it also allows $\{\texttt{cat}, \texttt{dog}\}$. Since our task is now

---

[1] For example, see the third postulate in Niemann (1983).

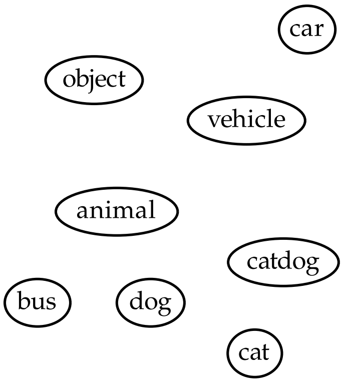


Figure 3.1.: A loose set of concepts, where mutual exclusivity does not hold.

too general, it needs to be restricted again, imposing a new structure on $\mathcal{Y}$. Formally, this is known as *structured output prediction* (*cf.* Shalev-Shwartz and Ben-David 2014, 198 sqq.), and we discuss methods that are formulated in this way in section 4.1.1. Our hierarchical classifier proposed in section 5.2 fits this definition as well.

In the following, we formally define the intuitions behind sets like in fig. 3.1, and how relations can be used to structure a label set. We then focus on different types of semantic relations and the knowledge represented therein.

3.1.1. Concepts

Before we can begin using concepts to structure our models and guide the development of our methods, we must first offer a definition. A concept is a *semantic entity*. In philosophy, a concept is also seen as a unit of thought. It has two main components, intension and extension (*cf.* Fitting 2020). The intension of a concept is the meaning behind it, its connotation or idea. The extension then specifies what *things* the concept applies to in the real world.

3.1.1.1. Formal Concept Analysis

In Wille (1992), the author offers a formal definition. The definition is based on set theory and is part of a larger theory named *formal concept analysis*. Concepts cannot exist in a vacuum. They need a space in which their extension and intension can be represented. In Wille's theory, this is called a *context* $\mathfrak{C} = (G, M, I)$, where:

- G is a set of objects ("Gegenstände"),
- M is a set of attributes ("Merkmale") and
- $I \subseteq G \times M$ is a binary relation. gIm means that the object g has the attribute m.

A concept is in this theory is a pair (A, B) with $A \subseteq G$ and $B \subseteq M$. The set of objects A is the *extent* (from extension) of the concept. Similarly, the set of attributes B is the *intent* of the concept. For consistency, it is required that the attributes B are shared by all objects in A and vice versa. The theory of formal concept analysis is developed further in Wille (2005). However, we consider this theory too complex for our applications. Hence, we simplify it in the following section.

3.1.1.2. Concepts in this Work

For the remainder of this thesis, we do not consider attributes to inform the intension of concepts. However, it should be noted that learning with attributes is an active research area and a great example of semantic knowledge integration. We further simplify our theory by defining both the extension and intension of concepts only over other concepts. Hence, a context is simply the set of all concepts $\mathcal{C}$. There are neither separate objects as extensions, nor are there attributes as intensions of concepts.

The context for the example in fig. 3.1 is the set:

$$\mathcal{C} = \{\texttt{object}, \texttt{vehicle}, \texttt{animal}, \texttt{car}, \texttt{bus}, \texttt{cat}, \texttt{dog}\}.$$

On its own, the context has no structure. We intuitively define the extension $E : \mathcal{C} \rightarrow \mathcal{P}(\mathcal{C})$ of a concept c as all concepts c applies to. For example:

$$E(\texttt{animal}) = \{\texttt{animal}, \texttt{dog}, \texttt{cat}\}.$$

In the same way, we can treat all concepts that apply to a concept c as its intension $I : \mathcal{C} \rightarrow \mathcal{P}(\mathcal{C})$, such that:

$$I(\texttt{animal}) = \{\texttt{object}, \texttt{animal}\},$$

which has an attribute-like interpretation.

A formal definition of intension and extension is provided as eqs. (3.1) and (3.2) in section 3.1.2.2.

The tuple $(\mathcal{C}, E, I)$ is a *taxonomy*. Clearly, the semantic knowledge necessary to give a structure to $\mathcal{C}$ is contained in E and I. In the following sections, we explore the sources and representations of such knowledge.

> Note that our use of the word concept for elements of the context, while incompatible with formal concept analysis (*cf.* Wille 1992), is common in the field of natural language processing (NLP) (*cf.* Harispe et al. 2015, 44 sq.) and also machine learning (*cf.* Silla and Freitas 2011).

3.1.2. Relations

Relations are a natural description of the extension and intension of concepts. We use them in following sections to represent concept hierarchies. However, they should first be defined formally.

3.1.2.1. Formal Definitions

In general, we use the term *relation* to mean a binary, homogenous relation. Thus, a relation R is a (non-strict) subset of the Cartesian product of a set X with itself: $R \subseteq X \times X$ (*cf.* Givant 2017, p. 1).

Let us first consider four special cases of a relation over X. There are (*cf.* ibid., p. 2):

- The *empty* relation $\emptyset$,
- the *universal* relation $X \times X$,
- the *identity* relation $id = \{(x, y) : x, y \in X \land x = y\}$ and
- the *diversity* relation $di = \{(x, y) : x, y \in X \land x \neq y\}$.

Relations are sets, and operations such as the complement, union, intersection and difference extend naturally. There are also binary operations specific to relations. The composition of two relations R and S, represented in this work as $\circ$, is of special interest. It is defined as (*cf.* ibid., 6 sq.):

$$R \circ S = \{(x, z) : \exists y : (x, y) \in R \land (y, z) \in S\} .$$

A relation $R \subseteq X \times X$ is (*cf.* ibid., p. 13):

- *Reflexive* if $id \subseteq R$, *i.e.*, every element relates to itself,
- *symmetric* if $(x, y) \in R \implies (y, x) \in R$,
- *transitive* if $(x, y) \in R \land (y, z) \in R \implies (x, z) \in R$,
- *antisymmetric* if $(x, y) \in R \land (y, x) \in R \implies x = y$.

We are mainly interested in relations that are transitive, reflexive and antisymmetric for the purpose of taxonomies. Relations with these properties define a *partial order* (*cf.* ibid., p. 13). There are also *strict* partial orders, where the relation is transitive, irreflexive and asymmetric. They are closely related, and many semantic relationships can be described in both manners.

With transitivity and the composition operation $\circ$, we can define two representations of relations that are used to transform taxonomies into graphs (see section 3.2.2). The *transitive closure* R^+ of a relation R is the minimal relation that is both transitive and contains R. If R^i is the relation obtained by composing R with itself i times, then $R^+ = \bigcup_{i=1}^{\infty} R^i$. The *transitive reduction* R^- of R is the smallest relation that has the same transitive closure as R (*cf.* ibid., 144 sqq.).

3.1.2.2. Semantic Relations

We now explore typical relations between semantic entities. Our main interest is in relations that can be used to build a taxonomy. However, there are non-taxonomic relations that also represent semantic knowledge.

The *hyponymy* relation is also known as the entailment, inclusion or *is-a* relation between nouns (*cf.* Brinton 2000, p. 135; Cruse 2002). It is the foundation for the methods and models in this thesis. It relates narrower with more general terms, *e.g.*, a `car` is a `vehicle`. In linguistics, the hyponymy relation is considered transitive, irreflexive and asymmetric, *i.e.*, it induces a strict partial order. However, we use it to describe the extension of a concept as proposed in section 3.1.1.2. Hence, we include *id* to form the reflexive and antisymmetric hyponymy relation $\leq_{\text{is-a}}$, *e.g.*, such that `dog` $\leq_{\text{is-a}}$ `dog`.

Definition 3.1 (Extension and intension)**.** With $\leq_{\text{is-a}}$, we formally define the extension $E : \mathcal{C} \to \mathcal{P}(\mathcal{C})$ of a concept c as:

$$E(c) = \{c' \in \mathcal{C} : c' \leq_{\text{is-a}} c\}\,, \tag{3.1}$$

and the intension $I : \mathcal{C} \to \mathcal{P}(\mathcal{C})$ as:

$$I(c) = \{c' \in \mathcal{C} : c \leq_{\text{is-a}} c'\}\,. \tag{3.2}$$

The inverse relation to hyponymy is called *hypernymy*.

Troponymy is a relationship between verbs. It is specific to WordNet (Miller 1995, see also section 3.3.1.1) and proposed in Fellbaum and Miller (1990). Troponymy describes the “manner” relationship, *e.g.*, sweeping is a manner of cleaning. Like hyponymy, it is a strict partial order, but on verbs instead of nouns.

A more practical semantic relation, *meronymy*, connects parts of something to the whole (Brinton 2000, p. 133). It is relevant for computer vision applications because it can have a spatial extent. For example, `wheel` is a part of `car` — not only in a semantic sense, but also spatially, and thus, visually. Datasets like Visual Genome (see Krishna et al. (2017) section 3.3.2.1) encode such relationships. Knowledge of meronymy can be used to improve generalization (*cf.* Sirakov et al. 2015).

Hyponymy, troponymy and meronymy all describe nested sets in a sense. This idea of hierarchy is explored in more detail in section 3.2. However, there is also a fundamentally different semantic relationship: *antonymy* (*cf.* Brinton 2000, p. 136). There are two kinds. The first kind, *binary* antonymy, relates complementary concepts such as `dead` and `alive`. The second kind, *non-binary* antonymy, connects extremes of a gradual concept, *e.g.*, `hot` vs. `cold`.

3.2. Concept Hierarchies

In section 3.1.2.2, we describe different semantic relations between concepts. The goal is to derive the extension and intension of concepts from these relations to ultimately build a taxonomy. We specifically focus on the hyponymy relation in this thesis. However, most of the methods and models apply to any relation with similar properties w.r.t. to the order they induce.

A useful perspective on a taxonomy of hyponyms and hypernyms is hierarchical. Concepts are arranged in a hierarchy of nested sets, where concepts subsume sets of other concepts. This *concept hierarchy* representation is possible because the underlying relation is a (partial) order.

Concept hierarchies are introduced in Lu (1997). The idea is more commonly known as *ontology* (*cf.* Lorhardus 1606; Neches et al. 1991). In the following, we discuss two ways of representing concept hierarchies. First, as relations and orders, using the theory introduced in section 3.1.2.1. Second, as graphs, for better algorithmic processing and improved visual understanding.

3.2.1. Concept Hierarchies from Relations

This section details the process of deriving a hierarchy from an underlying relation. We begin by addressing the requirements w.r.t. the properties of a relation. The idea of comparability is also explored.

3.2.1.1. Requirements

To build a hierarchy, we expect the relation to induce a partial order. Such a relation is reflexive, antisymmetric and transitive (*cf.* Givant 2017, p. 13). Transitivity is necessary for nesting, which is a fundamental property of hierarchies.

The hypernymy relation $\leq_{\text{is-a}}$, the focus of this thesis, is transitive. However, as mentioned in section 3.1.2.2, it is arguable whether it should be reflexive considering its original definition. While `dog` $\leq_{\text{is-a}}$ `dog` seems like a reasonable statement, the literal meaning of “hyponym” is “under-name”. It suggests a more strict, non-reflexive interpretation. In linguistics, the reflexive part of the relation even has a separate name: *autohyponymy* (*cf.* Gillon 1990).

For consistency with linguistics, we would have to consider hyponymy as inducing a strict partial order, *i.e.*, transitive, but irreflexive and asymmetric. However, we use the reflexive formulation as it is more practical for our purposes, and the derived intension and extension are unique for each concept. Furthermore, it allows for more concise descriptions in section 5.2.

3.2.1.2. Comparability

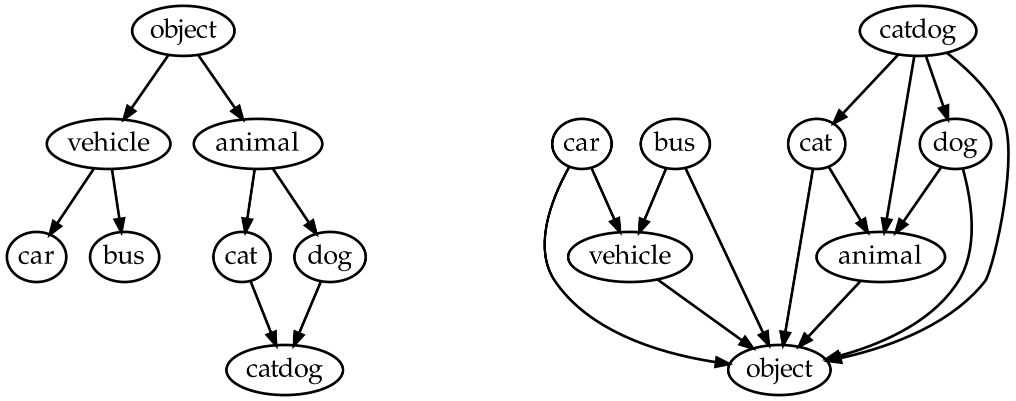
Relations can have varying degrees of completeness. For example, in a *total* relation R, every possible pair (x, y) of elements of the underlying set is *comparable*. That means that either $(x, y) \in R$ or $(y, x) \in R$ (*cf.* Givant 2017, p. 39). In an empty relation $\emptyset$, no two elements are comparable.

Hyponymy is a partial order, which means that some, but not all elements are comparable. For example, `dog` and `cat` are not hyponyms in either direction. Thus, it is not always possible to determine the most broad or narrow concept in a given set. However, when two concepts are not comparable, we can deduce that they are mutually exclusive in a sense. This type of semantic knowledge is used, *e.g.*, in Deng et al. (2014). Still, this does not always apply (see section 5.2.2).

Moreover, if a hyponymy relation were total, all concepts would be comparable, and no concepts would exclude each other. In other words, there would exist only one class.

3.2.2. Concept Hierarchies as Graphs

Graphs are a popular choice to represent concept hierarchies as they are visually approachable and many intuitive assumptions about their interpretation align with



(a) A typical graph representation of a concept hierarchy defined by a hyponymy relation.

(b) A "correct" graph representation of a concept hierarchy, where the edges are the hyponymy relation.

Figure 3.2.: Comparison between different graph representations of the same concept hierarchy.

hierarchies. Common operations on concept hierarchies are defined using graph theory, *e.g.*, semantic similarity in terms of distance in a graph (see section 6.1.2.2). This section details the basics of representing a concept hierarchy as a graph and also describes specific subtypes of graphs that are especially applicable.

3.2.2.1. Graph Representation

Formally, a *graph* G is a pair (V, E), where V is the set of vertices, or nodes, and $E \subseteq V \times V$ is the relation describing the edges, or connections between nodes (*cf.* Diestel 2017, p. 2).

In this work, we consider directed graphs (*cf.* ibid., p. 27), *i.e.*, E is not symmetric. See fig. 3.2a for an example of a directed graph. While this figure visualizes a concept hierarchy, the relation E is *not* transitive, and it is not a hyponymy relation.

What we visualize, and store in memory for algorithmic use, is the transitive reduction (see section 3.1.2.1). Transitivity is implied, and is expressed in the *directed reachability* relation over the graph. We can also derive the hyponymy relation using E^+, the transitive closure of the relation of edges E. However, there is another difference. The relationship is often inverted such that the notions of "parent" and "child" nodes align with the subsumption of concepts. Directed edges point away from the "root", or the most generic concept.

Another benefit of graphs is the concept of *connectedness*, where every node should have at least one edge connected to it (*cf.* ibid., p. 10). This is a requirement for most implementations of hierarchical classifiers (*cf.* Silla and Freitas 2011), while the stronger notation of connexity or totality in relations is not, and is also not compatible with hyponymy relations (see section 3.2.1.2). The relation equivalent to a connected graph that represents hyponymy is a *directed partial order*, where any subset of the underlying set must have an at least as or more general element in the whole set. Clearly, it is easier to express this requirement using graph theory.

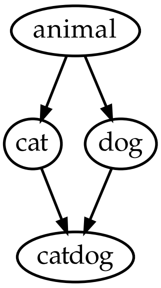


Figure 3.3.: A directed cycle in a graph which exhibits a “diamond” pattern.

3.2.2.2. Trees and Directed Acyclic Graphs

There exist two specific types of graphs that lend themselves to representing concept hierarchies. Both restrict the existence of cycles, *i.e.*, paths of length ≥ 3 where the first and last node are identical (*cf.* Diestel 2017, p. 8).

The first type is the *tree*, a connected acyclic graph (*cf.* ibid., p. 13). Trees are directed graphs, but no cycles can exist at all. The second good representation is the directed acyclic graph (DAG). Here, only *directed cycles* are forbidden. Undirected cycles, *e.g.*, “diamonds” can exist (see fig. 3.3). Diamonds are a term from object-oriented programming, where they represent multiple inheritance.

In practical applications, both types exist. On the one hand, the hyponymy relation of the popular WordNet database (Miller 1995) is a DAG. Biological taxonomy, on the other hand, has no cycles at all. Many hierarchical classification methods cannot process concept hierarchies that are DAGs (*cf.* Silla and Freitas 2011). The method we propose in section 5.2 can do so, but cannot assume mutual exclusivity of siblings as a consequence, which may have performance implications (see section 6.6.2).

There are theoretical advantages to restricting a concept hierarchy to a tree representation. For example, the length of the shortest path between two nodes in a tree is a metric. In a DAG, it is possible that the triangle inequality is violated (*cf.* Barz and Denzler 2019). Trees also have the advantage that all shortest paths are unique.

3.3. Semantic Knowledge Bases

Knowledge is a central theme of this thesis. While the actual definition of knowledge is subject to debate and extensive research (epistemology), we specifically mean the knowledge contained in a concept hierarchy.

In this section, we show where such semantic knowledge can come from. There are several types of knowledge bases. We start with lexical databases, which supply us with concept hierarchies.

Our work assumes that image datasets and semantic knowledge exist separately, but there are also datasets that combine images with enhanced semantics, which we describe in the following. We further mention subject-specific knowledge bases. These are of particular importance because they show that niche applications can also profit from knowledge integration. We also consider machine-readable knowledge bases with formalized semantics, which are crucial for cost-effective applications of machine learning integrated with semantic knowledge.

3.3.1. Lexical Databases

Lexical databases are one important source for concept hierarchies in this work. A lexical database is in essence a dictionary, but one that is accessible to algorithms. Such databases are often enhanced with semantic relations from which a hierarchy can be constructed (see also section 3.2.1).

3.3.1.1. WordNet

WordNet, presented in Miller (1995), is a crucial resource for research in computer linguistics and NLP. It is the de-facto standard lexical database and essential for this thesis. An online browser[2] can be used to explore the database, but there are also programmatic ways to access the data, *e.g.*, NLTK described in Loper and Bird (2002).

The WordNet database contains English words, *i.e.*, nouns, verbs, adjectives and adverbs. However, the units are not simply words in the lexical sense, but *synsets*. Synsets are unordered sets of synonyms, *i.e.*, lemmata that all represent identical concepts. For example, the synset `dog.n.01` contains the lemmata `dog`, `domestic dog`, `canis familiaris`. The lemma `dog`, however, appears in six other synsets, *e.g.*, as a mechanical part. To distinguish them, synsets are numbered, *e.g.*, such that the animal is `dog.n.01`, but the part is `dog.n.06`.

All relations in WordNet are between these synsets. This is ideal for our application. First, because different lemmata for the same concept are not important for visual recognition. And second, because the numbering of synsets avoids confusion when the same lemma has multiple concepts.

For nouns, WordNet offers hyponymy and meronymy relations. Verbs are enhanced by troponymy and adjectives have both direct and indirect antonyms. These correspond to binary and non-binary antonyms as described in section 3.1. There are also relations between different parts of speech. For example, the pertainymy relation links nouns with adjectives such that `criminal` pertains to `crime`.

Lexical databases such as WordNet that are augmented with relations are also known as *semantic networks*.

3.3.1.2. Multilingual WordNet Variants

The WordNet lexical database is of high quality, but limited scope. Most importantly, it is only available in English. However, several alternatives, extensions and combinations exist for other languages.

The BabelNet project, proposed in Navigli and Ponzetto (2012), is an effort to create a very-large-scale multilingual lexical database and semantic network. It aggregates multiple data sources to reduce or eliminate any human annotation effort. The current version, BabelNet “live”[3], uses at least 16 online sources that are constantly updated. It is able to cover 284 languages with this strategy. To align the languages, BabelNet

---

[2] `https://wordnet.princeton.edu/`

[3] `http://live.babelnet.org/`

relies on translations from Wikipedia, Wikidata and other multilingual knowledge bases.

While BabelNet attempts to cover almost all possible languages, there are also smaller-scale efforts that focus on individual languages. MultiWordNet, presented in Pianta, Bentivogli, and Girardi (2002), is an example of such a project. Its goal is to provide an Italian version of WordNet that is strictly aligned to the English counterpart. Strict alignment means that all semantic relations are correctly transferable from the original WordNet. There are also specific annotation for concepts that have different extensions (see section 3.1.1) in either language, and lexical gaps.

The *Global WordNet Association* coordinates efforts on multilingual WordNet developments. Their main goals are the standardization of representations and the sharing of data between individual projects. The platform also provides guidelines for building further WordNets in new languages. A summary can be found in Pease, Fellbaum, and Vossen (2008).

3.3.2. Visual-Semantic Datasets

Technically speaking, most computer vision datasets (see, *e.g.*, section 6.2.1.1) combine some forms of visuals and semantics, typically in the form of images and class labels. Here, we explore datasets where semantics are enhanced, *i.e.*, contain more information or have a more complex representation than simple class labels or bounding boxes. Label representations include complex scene descriptions, captions and question-answer pairs.

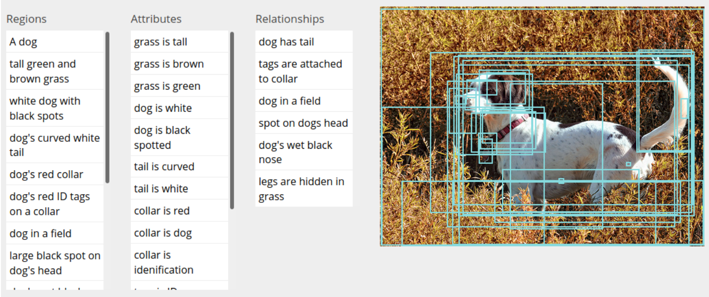
It should be noted that datasets based on ImageNet (see section 6.2.1.1), while not described in detail here, are also semantically enhanced because all labels are synsets taken directly from WordNet (section 3.3.1.1).

3.3.2.1. Visual Genome

Visual Genome, originally presented in Krishna et al. (2017), is probably the dataset with the most semantically complex annotations. For each single image, it encodes objects with semantic and visual relationships, attributes and bounding boxes for each part. An example image is shown in fig. 3.4, where the large amount of bounding boxes is clearly visible.

While the annotation complexity is impressive, the utility is further improved by the feature that all semantic units in all annotations are WordNet synsets. This allows one to take advantage of additional semantic knowledge from WordNet. The formal representation is a *scene graph*. It is similar to a knowledge graph (see section 3.3.3.2) in that it consists of triples of subject, predicate and object. Each individual element is associated with a region in the image by a bounding box.

The dataset consists of 108 077 images. In total, there are approximately 2 300 000 relationships encoded, connecting approx. 3 800 000 objects and also approx. 2 300 000 attributes.

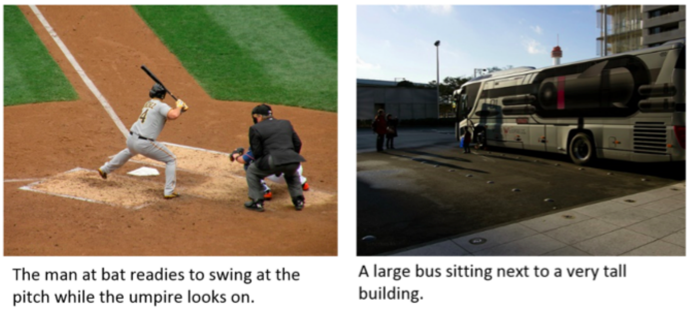


Regions
A dog
tall green and brown grass
white dog with black spots
dog's curved white tail
dog's red collar
dog's red ID tags on a collar
dog in a field
large black spot on dog's head

Attributes
grass is tall
grass is brown
grass is green
dog is white
dog is black spotted
tail is curved
tail is white
collar is red
collar is dog
collar is idenification

Relationships
dog has tail
tags are attached to collar
dog in a field
spot on dogs head
dog's wet black nose
legs are hidden in grass

Figure 3.4.: An example from the Visual Genome dataset. Image captured from `https://visualgenome.org/VGViz/explore?query=human%20behind%20dog` on February 10th, 2021.



The man at bat readies to swing at the pitch while the umpire looks on.

A large bus sitting next to a very tall building.

Figure 3.5.: An example of image captions from the Microsoft COCO dataset. Image captured from `https://cocodataset.org/#captions-2015` on February 11th, 2021.

3.3.2.2. Image Captioning

While scene graphs in Visual Genome (see section 3.3.2.1) allow for an exceptional degree of scene understanding, such annotations are very expensive and time-consuming. Image captions are a less informative, but still semantically rich type of label, where annotators are simply asked to describe the scene in words. Processing image captions involves NLP and evaluation schemes can be very complex and difficult (*cf.* Chen et al. 2015b).

Microsoft COCO (Common Objects in Context, *cf.* Lin et al. 2014) is mainly an object detection dataset. Objects are not just annotated with class labels, but as individual instances. Segmentation maps are also provided per instance for visually fine-grained scene understanding. On the semantic side, there are five image captions available for each of the approximately 328 000 images. Figure 3.5 shows two example captions.

Natural images contain not only objects, but occasionally also text. There are datasets that combine image captioning with optical character recognition (OCR), such that image captions also describe any legible text. TextCaps (*cf.* Sidorov et al. 2020) relates written text in images to objects and also offers interpretations in captions, such as signs or public information displays. COCO-Text (*cf.* Veit et al. 2016) focuses only on the text in images. It differentiates between hand-written and machine-printed text for a more challenging recognition task.

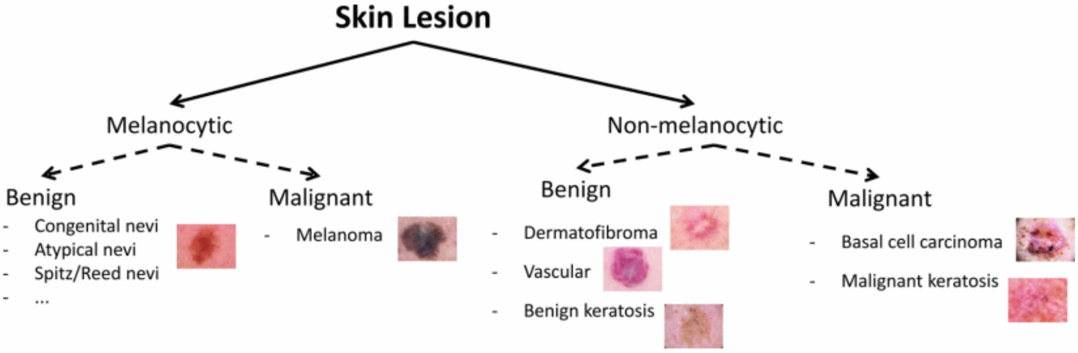


Figure 3.6.: A hierarchy of skin lesions. Figure taken from Barata, Marques, and Celebi (2019).

There is an effort by Google, Conceptual Captions (*cf.* Sharma et al. 2018), to automatically generate a very large dataset of more than 3 000 000 images. Instead of human annotations, they use image captions from websites and perform extensive filtering. There is a bespoke simplification step called "hypernymization" that makes the often very specific image captions more general.

3.3.2.3. Visual Question Answering

Visual question answering (VQA) is a complex visual-semantic machine learning task. The input to such a system is an image with an associated question, while the answer has to be predicted. This requires an understanding of the scene as well as the semantics of the question, leading to interesting challenges in model design. For example, consider a picture of a street and the question "can you park here?" as well as the knowledge required to answer it.

The eponymous VQA dataset presented in Antol et al. (2015) provides such data. Images are taken directly from MSCOCO (see section 3.3.2.2). The authors collect 760 000 questions from humans. To avoid challenges (*e.g.*, as in Chen et al. 2015b) in evaluating the correctness of answers, they also collect ten answers for each individual question. A predicted answer is then considered correct if it matches at least three of the human answers.

Further visual-semantic datasets include Fashionpedia (Jia et al. 2019) and VisualSem (Alberts et al. 2020).

3.3.3. Domain-Specific Knowledge Bases and the Semantic Web

WordNet is a helpful resource for general object recognition, as evidenced by its use in the construction of ImageNet (see section 6.2.1.1). However, industrial, medical or research applications have separate terminologies, such that we cannot use WordNet to build a concept hierarchy. For example, biology has its own taxonomy. In medicine, there are many hierarchies, *e.g.*, of skin lesions (*cf.* Barata, Marques, and Celebi 2019) that can be used. We explore potential domain-specific data sources in this section and also consider the formalization of such knowledge.

3.3.3.1. Biological

This thesis contains several biological applications, specifically in the field of biodiversity (see sections 7.3 and 7.4). In biology, the central hierarchy comes from the Linnean taxonomy system (*cf.* Linnæus 1758).

An important resource for taxonomic information is WikiSpecies[4]. It is represented as RDF data (see section 3.3.3.2) in the Wikidata[5] project, and as such, can be accessed with structured queries in the SPARQL language. We use it in this thesis for experiments that require knowledge of relations between species, *e.g.*, in section 6.5.

WikiSpecies is also cross-referenced with other databases, including the Catalogue of Life[6]. The catalogue is a meta-database combining 186 individual taxonomic databases. Hence, there is a high likelihood of finding and relating even the most obscure species.

While biological taxonomies are easily found in a machine-accessible format, there are many areas in biology where knowledge is mainly represented in natural language. Bioschemas (*cf.* Gray et al. 2017) is a recent attempt to formalize more areas by offering a unified markup language and schemas for life sciences.

3.3.3.2. Semantic Web

There are two main ways of accessing knowledge represented in a machine-readable fashion. The first is to apply NLP methods to process documents like scientific publications, textbooks and manuals. The second is to store any knowledge in a machine-readable format is the first place. “Semantic Web” is an umbrella term for technologies that aid in the second approach. They are also the technical foundation of databases like Wikidata.

An early semantic web technology is Resource Description Framework (RDF), proposed In Brickley and Guha (2014). It is a graph-based formalization of semantics. An RDF model consists of a number of triples with subject, predicate and object. Such a triplet is called a statement, and they are connected together to form a *knowledge graph*. For example, consider this excerpt from the WikiSpecies database (see also section 3.3.3.1):

> Subject: `Michotamia aurata` (Q1313312)
> Predicate: `instance of` (P31)
> Object: `taxon` (Q16521).
> —
> Subject: `Michotamia aurata` (Q1313312)
> Predicate: `parent taxon` (P171)
> Object: `Michotamia` (Q14510436).

Similar to WordNet, each human-readable term like `instance of` has a unique identifier (P31) to mitigate potential overlaps.

---

[4] `https://species.wikimedia.org/`

[5] `https://www.wikidata.org/`

[6] `https://www.catalogueoflife.org/`

OWL, the Web Ontology Language (*cf.* W3C OWL Working Group 2012), is a generalization of the concepts in RDF. It allows for more complex expressions. Taxonomies are explicitly modeled by considering classes, properties and instances as fundamental building blocks.

The *Linguistic Linked Open Data* platform integrates linguistic knowledge from a variety of sources using semantic web technologies to allow for easier sharing of data and automatic access. Their positions are stated in Chiarcos, Hellmann, and Nordhoff (2011). It is a promising development for semantic knowledge integration because it aims to make even more knowledge accessible to methods like ours proposed in section 5.2.

4. Literature Overview

This chapter serves as a broad overview of literature that is relevant to this thesis. The related work can be divided into roughly two categories. First, we discuss methods of semantic knowledge integration into classification tasks. The respective knowledge is almost always a concept hierarchy, however, we also consider other semantic information such as attributes. In addition to methods of hierarchical classification, we review several exploratory and analysis works. The second category concerns problem formulations other than general classification tasks. Here, we focus on problems that stand to benefit from semantic knowledge integration. We discuss literature on imprecise data, several alternatives to (fully) supervised learning, and natural language processing.

4.1. Semantic Knowledge Integration From Concept Hierarchies

In this section, we review related work that proposes methods of hierarchical classification, or more generally, methods that incorporate semantic knowledge in the form of a concept hierarchy to improve a classification task. The term hierarchical classification subsumes a variety of different tasks and methods (Silla and Freitas 2011). All of them involve a concept hierarchy at some point to either constrain or aid in the learning of a classifier. In the following, we point out important distinctions and give an overview of relevant work in this field. We first list approaches where a hierarchy is used to structure the classifier itself, *e.g.*, to build connections in a neural network. Afterwards, we discuss the mutually related topics of embeddings and metric learning. Finally, we point out non-semantic hierarchies one may encounter in computer vision and a variety of interesting analyses.

4.1.1. Model Structure

A major category of hierarchical classifiers are *structural* models. We consider a method structural if the construction of the model itself is informed by a concept hierarchy. Examples are special layers, or decision trees that are isomorphic to a hierarchy, *i.e.*, where the model's architecture is hierarchical. We also list probabilistic graphical models where the dependency graph is built from a concept hierarchy. Methods in which a concept hierarchy is used to construct to a loss function are also included for completeness.

4.1.1.1. Hierarchical Architecture

An early structural method is presented in Marszalek and Schmid (2007), where SVMs are nested to represent a given concept hierarchy. This improves both accuracy and runtime compared to a contemporary one-vs-all SVM. In addition to hyponymy, the work also explores applications of integrating meronymy and holonymy relations (see section 3.1.2.2).

Boosting is a meta-learning method, where multiple "weak" classifiers with access to a subset of features are aggregated into one hypothesis. In Fan, Gao, and Luo (2008), SVMs are aggregated in a hierarchical boosting approach.

Hwang (2013) proposes a collection of structural hierarchical classification methods, *e.g.*, the *Tree of Metrics*. The author also discusses the separation of knowledge between training data and concept hierarchy, considering the latter "external", as we do in this thesis.

Furthermore, the structure of a concept hierarchy can guide the construction of a neural network. Many approaches consider a simplified hierarchy of classes and superclasses, *i.e.*, with only two levels. In Ahmed, Baig, and Torresani (2016), specialist and generalist networks informed by both semantic and visual hierarchies (see section 4.1.4) are proposed. Goo et al. (2016) considers hierarchical pooling layers to represent generalization and specialization of features between classes and superclasses. Knowledge graphs, textual descriptions of concepts and further knowledge are integrated in Zhang, Lertvittayakumjorn, and Guo (2019), while still only two hierarchy levels are considered.

Instead of restricting the concept hierarchy to two levels, one can assign one layer of the neural network directly to one level in the concept hierarchy. For example, Zhu and Bain (2017) propose one layer with a softmax activation function (see section 2.4.1.3) per level. Their method is also used in Zhang, Mou, and Xie (2020) to integrate semantic knowledge for image generation. Similar one-layer-per-level approaches are discussed in Chen et al. (2018) and La Grassa, Gallo, and Landro (2021). Roy, Panda, and Roy (2020) uses a "Tree-CNN" not to improve accuracy, but to increase the complexity as new concepts are encountered in an incremental learning scenario (see section 7.1.2).

In Li et al. (2021), the authors propose a generalization of hierarchical classification that allows for multiple hierarchies. The model differentiates between concepts that occur in multiple hierarchies ("common subclasses") and concepts that are restricted to one of the hierarchies ("heterogeneous superclasses"). Each set of concepts is assigned a separate branch of the underlying convolutional neural network.

> The concept of "level" only applies to concept hierarchies that are trees (see section 3.2.2.2). Hence, the aforementioned methods are not all compatible with DAGs.

4.1.1.2. Probabilistic Graphical Models

With the knowledge of a concept hierarchy and additional assumptions, the probabilities of certain combinations of concepts can be inferred (*cf.* section 5.2). In the following, we discuss a selection of methods that manipulate probabilities to integrate knowledge.

Taskar, Guestrin, and Koller (2004) lays out a general framework to combine probabilistic graphical models with SVMs. While kernel-based SVMs contribute their capability of learning in high-dimensional feature spaces (see section 2.3.3.1), the probabilistic graphical models can exploit the problem structure, *e.g.*, as derived from a concept hierarchy. A similar approach is discussed in Chen et al. (2015a), where probabilistic graphical models are combined with deep neural networks.

In Deng et al. (2010), the authors explore the special requirements of classification tasks with tens of thousands of classes. A key insight is that not all confusions, *i.e.*, predictions that differ from the ground truth, are of equal consequence. They propose a misclassification cost based on semantic distance (*cf.* section 5.1.1.1) and derive a probabilistic Bayes classifier to minimize this objective. Concept hierarchies are used to derive prior probabilities for a deep neural network's weights in Srivastava and Salakhutdinov (2013). The (posterior) knowledge in the model is updated by modifying the concept hierarchy, *e.g.*, by removing relations, during training. This Bayesian approach combines the strengths of prior knowledge and learning from data.

A conditional random field that combines knowledge from a concept hierarchy as well as attributes is proposed in Samplawski et al. (2019). Instances of knowledge are modeled as expressions of propositional and relational logic. The conditional random field is combined with a deep neural network.

While the aforementioned approaches require special models and learning processes, the method proposed in Karthik et al. (2021) can be applied to an already existing and trained deep neural network. A post-processing step adjusts the confidences scores given by the last layer to improve the quality of mispredictions, *i.e.*, lower their semantic distance. This is also known as making "better mistakes" (*cf.* Bertinetto et al. 2020).

4.1.1.3. Regularization and Loss Functions

Without affecting the structure of a classifier, or explicit modeling of probabilities, there is another possibility to integrate knowledge from concept hierarchies into a machine learning system. Consider Hypotheses obtained from ERM, which are the result of optimizing a loss function. In section 2.2.1, we already discuss regularization, where an additional term to the loss function is used to improve generalizability. Similarly, the loss function can be modified to integrate semantic knowledge.

Fergus et al. (2010) proposes "Label Sharing", where a one-hot encoded representation of the ground truth labels is modified to induce knowledge transfer between concepts. The authors define an affinity matrix which indicates the degree to which two concepts are related semantically. This matrix is only a slight deviation from the identity matrix. Each one-hot encoded label is then transformed by the affinity matrix and an adapted cross-entropy loss function (see section 2.4.1.4) enforces the prediction of the modified labels, which are interpreted in terms of multi-label classification.

A loss function specifically tailored to semantically imprecise data is derived in McAuley, Ramisa, and Caetano (2013). The derivation is by way of an evaluation metric for imprecise predictions (see section 5.2.3.1).

Xie et al. (2015) introduces hyperclasses, which are identical to the previously discussed superclasses in a two-level hierarchy. Assuming that there is a comparatively large amount of training data available that is annotated with hyperclasses, a neural network is first trained only on hyperclasses. A second neural network then solves the actual classification task, while a term in the loss function enforces an alignment between both networks to induce knowledge transfer.

In Chen et al. (2019), a loss function component is added specifically for the concepts which should be predicted with low probabilities. In a conventional classification task, these complementary concepts are simply all concepts other than the ground truth. However, when a concept hierarchy is available, a higher priority can be assigned to more general complementary concept, which in turn "neutralize" their hyponyms. This combined loss function is called a "hierarchical complement objective".

Goyal and Ghosh (2020) is a unique approach. While other loss functions depend only on the ground truth and the predictions, this method builds a *curriculum*, *i.e.*, a loss function that depends on the current number of optimization iterations. The concept hierarchy is used to order the training data, which results in a significant accuracy increase, and has further theoretical benefits. A similar approach is proposed in Stretcu et al. (2021).

4.1.2. Embeddings

Another category of approaches for hierarchical knowledge integration into classification tasks are those based on *embeddings*. An embedding is a high-dimensional vector space $\mathcal{E}$ equipped with two maps, one from the domain set $\mathcal{X}$ to $\mathcal{E}$ and one from the label set $\mathcal{Y}$ to $\mathcal{E}$. These maps can be either fixed or learned, *e.g.*, by deep neural networks. Predictions are inferred by applying the first map from $\mathcal{X}$ to $\mathcal{E}$, and then the inverse of the second map. If the second map is not invertible, a nearest neighbor approach in $\mathcal{E}$ can be applied.

Frome et al. (2013) combines a CNN for image classification with a skip-gram text model that learns a high-dimensional vector representation of words. Both models are initialized separately, the vision model on ImageNet-1k (see section 6.2.1.1) and the text model with data from Wikipedia. The vision model is then fine-tuned to minimize a similarity metric w.r.t. the vector representation of the labels. This combination results in qualitatively improved mispredictions (see also Bertinetto et al. 2020) and enables zero-shot learning (see section 4.2.4.3).

Attributes and superclasses are combined in a label embedding in Hwang and Sigal (2014). The superclasses reside in the same vector space and are enforced to be linear combinations of their constituent classes. A similar idea is explored in Norouzi et al. (2014), where an embedding space is constructed from convex combinations of one-hot classifier outputs.

Sun et al. (2017b) allows a CNN to learn an embedding freely in order to exploit possible relations between labels. It is observed that the learned relations reflect semantic similarities (see section 6.1.2.2) even though the learning process has no access to the concept hierarchy (see also section 4.1.5).

While most methods capitalize on the nested subset aspect of concept hierarchies, Vendrov et al. (2016) considers the partial order represented by a hyponymy relation (see section 3.1.2.1). The authors propose a label embedding into a real vector space that, with the reverse product order on its elements, is order isomorphic to the hyponymy relation.

Embeddings that integrate both similarity and order at the same time are proposed in Nickel and Kiela (2017) and further generalized in Ganea, Becigneul, and Hofmann (2018) and Dhall et al. (2020). Nested volumes (cones) in the embedding space represent concepts such that the nesting matches the concept hierarchy. Crucially, the hyperbolic geometry enables a volume hierarchy that matches a DAG, *i.e.*, concepts with multiple hypernyms.

Barz and Denzler (2019) considers a fixed label embedding, such that Euclidean distances between concepts correspond (inversely) to semantic similarity. The embedding is used for classification and also to provide more reasonable results in content-based image retrieval (see section 4.2.3). A similar approach is developed in Jayathilaka, Mu, and Sattler (2020). Learned (instead of fixed) embeddings reflecting semantic similarities are proposed in Narayana et al. (2019) and Arponen and Bishop (2019).

4.1.3. Metric Learning

Metric learning methods consider a vector space similar to embedding methods, or more generally, a manifold. However, there is no explicit mapping from and to of this space. Instead, metric learning approaches learn and predict distances, or metrics, between two domain points directly. To perform a classification task using such an approach, the learned distance can be used to determine the nearest neighbor in the training data.

In Verma et al. (2012a), a nearest neighbor classifier is equipped with a matrix that scales the euclidean feature distance according to a concept hierarchy. It is shown to improve classification accuracy and, as a side effect, can perform zero-shot learning (see section 4.2.4.3).

A conventional *triplet loss* (*cf.* Schroff, Kalenichenko, and Philbin 2015) considers a triplet of anchor, positive, and negative examples. The positive example should be close to the anchor, while the negative example should be at least a certain *margin* away. Zhang et al. (2016) proposes a hierarchical triplet loss, where the margin is dependent upon the semantic distance (see section 6.1.2.2) between anchor and negative example. Similar approaches are described in Faghri et al. (2018), Tonioni and Di Stefano (2019), and Lin, Gao, and Li (2020). In Wang et al. (2017a), the distance scale is removed from the triplet loss in favor of enforcing certain angles, which are determined by a concept hierarchy. Proença, Yaghoubi, and Alirezazadeh (2020) generalizes the triplet loss to a *quadruplet loss*, enforcing semantic distances between two independent pairs.

Sampling three examples to compute the triplet loss is not trivial and typically involves complex strategies such as *hard negative mining* (*cf.* Schroff, Kalenichenko, and Philbin 2015). There are works that relax this requirement, including the aforementioned Proença, Yaghoubi, and Alirezazadeh (2020), which can utilize random samples. He et al. (2021) uses a concept hierarchy to guide the sampling and determine hard triplets more efficiently. A method that requires no sampling at all is proposed in Qian et al. (2019).

Sanakoyeu et al. (2019) splits the metric learning problem into several smaller sub-problems along domain points as determined by a concept hierarchy. The less complex sub-problems result in faster convergence and better generalization.

4.1.4. Non-Semantic Hierarchies

The hierarchies used in this thesis almost always represent *semantic* knowledge (see section 3.2). They are nested sets of concepts informed by a hyponymy relation. In this section, we explore other hierarchical structures that appear in computer vision tasks, which are mostly *visual* and *spatial* hierarchies. These hierarchies are typically learned or discovered and not supplied as external knowledge.

Deng et al. (2011) learns a tree of classifiers to maximize both accuracy and efficiency. The sole purpose of this tree is to divide a large classification problem into smaller subproblems.

HD-CNN proposed in Yan et al. (2015) a further example of learning a hierarchy from data. “Coarse” categories are defined from a spectral clustering of the confusion matrix. HD-CNN specifically allows overlapping coarse categories, *i.e.*, hierarchies that are not trees, but DAGs. The clustering can only rely on visual properties because the classifier is not aware of semantics. A similar approach is discussed in Fan et al. (2015), where a hierarchy is constructed by clustering visual features.

Ahmed, Baig, and Torresani (2016) combines a visual hierarchy and a concept hierarchy in a two-level classification method (see also section 4.1.1.1).

In Fan et al. (2017), visual features of different granularities or levels of abstraction are acquired by extracting the activations from different layers of a CNN. These visual features are then used to build a tree classifier similar to Fan et al. (2015), which is trained jointly with a CNN. Zhao et al. (2018) presents a comparable method.

A visual hierarchy of images is constructed based on visual feature distances in Ge (2018). Based on this hierarchy, the margins for a triplet loss are computed (see section 4.1.3). Milbich et al. (2020) discovers “reliable” pairwise relations between images based on unsupervised clustering of visual features. The result is a feature representation learned without labels that can be used in downstream tasks, *e.g.*, classification.

Knowledge about the spatial relation of components is used in Sirakov et al. (2015) to accurately identify firearms in X-ray scans even if they are partly occluded or disassembled. Mo et al. (2019) proposes a three-dimensional point cloud dataset for segmentation that exposes a nested spatial structure (“parts”).

4.1.5. Analysis and Exploratory Works

This section briefly lays out several interesting observations and connections related to concept hierarchies and computer vision tasks.

Bilal et al. (2018) hypothesizes that CNNs naturally learn a concept hierarchy from training data. The authors show that the confusion probabilities between concepts exhibit a correlation with the concepts' semantic similarity, which corroborates the findings of Deng et al. (2010). We offer a related investigation in section 6.6.1. They further demonstrate that the depth of a layer in a CNN corresponds to the depth in a concept hierarchy at which the features from the layer help discriminate.

In Seeland et al. (2019), the authors test whether visual commonalities between genera or families of plants allow for successful classification of unseen species into the correct family or genus (see also section 4.2.4.3). They report that classifying new unseen species is possible, but at a significantly lower accuracy than known species.

Bertinetto et al. (2020) asserts that hierarchical classification methods "make better mistakes" than conventional classifiers that have no access to semantic knowledge. The mispredictions of hierarchical methods are "better" in that they are semantically more similar to the ground truth labels than mispredictions by non-hierarchical methods. This finding somewhat puts the results of Bilal et al. (2018) into perspective and suggests that hierarchical methods still have a knowledge advantage compared to ordinary classifiers, which we quantify in section 6.6.1.

4.2. Problem Formulations

In this section, we explore different formulations of machine learning problems which stand to benefit from semantic knowledge integration using concept hierarchies. These include variations of the classification task and also problems that do not directly involve predictions or even labels.

4.2.1. Imprecise Data

The notion of imprecise data (see section 5.1) has various competing definitions. For example, they differ in which aspect of the data is imprecise, and in which manner. Some also consider imprecise predictions (see section 5.2.3.1) while others do not.

Qin and Lawry (2004) is an early work in the field. It defines imprecise concepts as "fuzzy sets", *i.e.*, sets where membership is probabilistic. These sets are used in the *computing with words* framework (proposed in Lawry 2001) to solve natural language processing tasks (see also section 4.2.5).

Imprecision in multi-label classification is discussed in Younes, Abdallah, and Denœux (2010). The authors consider annotations which are missing a fraction of the labels as imprecise, regardless of the semantics. They propose a modified k-nearest-neighbor classifier to solve the problem.

Deng et al. (2012) discusses semantically imprecise predictions. They are used to improve accuracy by trading off precision, which the authors call specificity (see section 5.1.1.2). Their algorithm "DARTS" can optimize this trade-off to achieve any given degree of accuracy. A similar trade-off is considered in Wu et al. (2020), where the precision of prediction is reduced for concepts that are under-represented in the training data. Davis et al. (2021) and Wang et al. (2021) optimize this trade-off in a probabilistic framework. In McAuley, Ramisa, and Caetano (2013), an evaluation metric that allows for such imprecise predictions is developed. This metric is then used to derive a corresponding loss function.

Chang et al. (2021) discusses semantic imprecision as a result of varying expertise or preference between annotators. Their assumptions are similar to our earlier work Brust, Barz, and Denzler (2021b), but less detailed (see also section 5.1.1.1). An even simpler model of label imprecision is proposed in Deng et al. (2014).

4.2.2. Fine-Grained and Large-Scale Classification

This section discusses classification tasks that are not special in any fundamental way, but nevertheless require bespoke methods. These include *large-scale* classification, where the number of concepts is too large to apply a simple mutual exclusion model, *e.g.*, softmax (see section 2.4.1.3). Another concern is the distribution of training data, which may over- or underrepresent certain concepts. Finally, we consider *fine-grained* classification tasks that require discriminating between very similar concepts.

Fergus et al. (2010) seeks to transfer knowledge between concepts such that less well represented concepts can still be learned reliably (see section 4.1.1.3 for a more detailed description). The method is proposed for learning with “many categories”, which in this case is validated on a dataset with 75 000 concepts. In Deng et al. (2010), a variant of ImageNet (see section 6.2.1.1) with over 10 000 concepts is used to discover semantic interdependencies in concepts. The same dataset is applied in Fan et al. (2017) and Zhao et al. (2018). These works are described in more detail in section 4.1.4.

The special properties of *long-tailed* classification problems are considered in Wu et al. (2020) (see also section 4.2.1). In this setting, few concepts are overrepresented, and most concepts are underrepresented. For example, this is the case when the training data distribution follows a power law.

Xie et al. (2015), which we elaborate on in section 4.1.1.3, applies the proposed method to fine-grained classification. The metric learning-based approaches in Zhang et al. (2016) He et al. (2021) are targeted towards the same application. These methods are detailed in section 4.1.3. A structural approach is proposed in Chen et al. (2018) (see also section 4.1.1.1).

Experimental data for fine-grained classification includes the “Stanford Cars” dataset proposed in Krause et al. (2013), which considers 196 models of cars. “Stanford Dogs” is a further benchmark, where 120 breeds of dogs are extracted from the ImageNet database. Biological data is of particular interest because species are often separated by small details, which provides a challenging classification task. The Caltech-USCD Birds 200 benchmark (CUB-200, *cf*. Welinder et al. 2010) and the North American Birds dataset (NABirds, *cf*. Van Horn et al. 2015) are important entries in the field as well. iNaturalist, described in Van Horn et al. (2018), is accompanied by a yearly challenge, and an example of a fine-grained, large-scale (with more than 5000 species) *and* long-tailed dataset. iWildCam, described in Beery, Cole, and Gjoka (2020), presents a similar yearly challenge of fine-grained visual recognition.

In Barz and Denzler (2021b), the authors present a fine-grained dataset of church images with the goal of classifying the respective architectural style. It exhibits several challenging properties, including few examples, high intra-class variance, and imbalanced classes. There is an included class hierarchy and a fraction of imprecise labels. However, due to the timing of its release two weeks before the submission of this thesis, it is not included in chapter 6.

Technical details concerning the aforementioned datasets can be found in sections 6.2.1.1 and 6.2.1.2.

4.2.3. Image Retrieval

In *content-based image retrieval*, a user provides a query, or reference image. The goal is to select images from a database that have similar content. Since visual similarity is often not what users are looking for, several strategies to integrate semantic knowledge are developed.

The following works all consider image retrieval as a metric learning problem (see section 4.1.3). A hypothesis has two inputs, the query image and a candidate image. The output is a predicted similarity score, by which all candidates are sorted.

Wang et al. (2017a), Faghri et al. (2018), Ge (2018), and Barz and Denzler (2019) all use measures of semantic similarity or distance (see section 6.1.2.2) derived from a concept hierarchy to inform the learned metric. By contrast, Sanakoyeu et al. (2019) utilizes a concept hierarchy to structure the learning process for better performance. Arponen and Bishop (2019) is a *hashing* method which computes low-dimensional representations of images. The distance between these hashes then corresponds to a semantic distance.

4.2.4. Alternatives to Supervised Learning

In section 2.1.4, we discuss supervised learning and its relation to other problem formulations such as unsupervised learning, semi-supervised learning and self-supervised learning. This section explores literature on the topic of supervision itself, as well as several methods that implement alternatives to supervised learning.

Damen and Wray (2020) formalizes the concept of supervision. The authors propose a three-dimensional scale by which the “supervision level” of a task can be measured. The three dimensions of the scale concern the extent, or coverage, of the training data, whether any pre-training is involved (see section 6.2.2.2), and the extent, or coverage, of the labels w.r.t. the training data.

In Geirhos et al. (2020), the processes by which humans, supervised learning methods and self-supervised learning methods produce predictions are compared. Interestingly, the inductive biases used by each appear to be similar, resulting in a preference of certain features, *e.g.*, textures.

Wei et al. (2020) states that labels should be preserved even when the domain changes in a pre-training and fine-tuning process (see section 6.2.2.2). They conclude that optimizing w.r.t. any discriminative task, even an unrelated one, results in better features than a typical fine-tuning step which only considers the “new” task. As such, the process is similar to the incremental learning method proposed in Käding et al. (2016b), which is further detailed in section 7.1.2.

4.2.4.1. Unsupervised Learning

For unsupervised learning (see section 2.1.4.1), concept hierarchies are of little consequence as there are no labels to impose a structure upon. Nevertheless, there is work exploiting (visual) relations between images, as discussed in section 4.1.4. Milbich et al. (2020) assumes that any collection of images contains useful and salient pairwise relations. The authors propose an unsupervised clustering algorithm, which learns these pairwise relations to produce a reliable feature representation without requiring any labels.

4.2.4.2. Weakly and Semi-Supervised Learning

Weakly supervised learning (see section 2.1.4.3) assumes that the quality of some or all of the labels in the training data is diminished in some aspect (*cf.* Zhou 2017). For example, it can be extremely noisy or imprecise (see section 5.1). Semi-supervised learning (see section 2.1.4.2) can be considered an extreme case, where labels are absent for a fraction of the training data.

In Dawson and Polikar (2021), the authors discuss the differences between honest mistakes and malicious labeling, which might occur in citizen science scenarios. They also propose a filtering method based on their insights.

Dehghani et al. (2017) considers learning from a mixture of strongly and weakly labeled (noisy) training data. A "confidence network" predicts the expected reliability of each label, which is then used as a weight for optimization. Labels that appear incorrect to the confidence network have smaller overall influence on the training, resulting in improved accuracy.

For a similar task, Cheng et al. (2020) exploits a concept hierarchy to obtain individual weights for elements of the potentially noisy training data. Images are embedded in a high-dimensional feature space, where distances conform to a semantic distance in the concept hierarchy. Comparing a prototype associated with the label to the actual image features results in a confidence score.

Yang et al. (2020) applies a word embedding method based on a concept hierarchy which compares metadata stored alongside labels to potentially correct them. A metric learning approach based on a semantically ranked triplet loss is used in Lin, Gao, and Li (2020) to perform semi-supervised learning.

4.2.4.3. Few-Shot and Zero-Shot Learning

Zero-shot learning is a task which requires a model to generalize not only to new examples, but also to new concepts that are not represented in the training data. If there are examples available, but only a small amount, the problem is considered *few-shot learning*.

Srivastava and Salakhutdinov (2013) tackles few-shot learning by providing an initialization for a CNN derived from a concept hierarchy (see section 4.1.1.2). Knowing the relations between concepts should enable and guide the transfer of knowledge from comparatively overrepresented concepts.

Zero-shot learning is impossible without domain knowledge in addition to the training data. However, domain knowledge can be supplied in various forms, including concept hierarchies, attributes, word embeddings derived from large corpora as well as textual descriptions of concepts. Verma et al. (2012a) use a concept hierarchy to learn a linear metric enforcing semantic distances in the feature spaces. The nearest-neighbor classifier can locate an unseen concept in the concept hierarchy, and even concepts that are not part of the learned concept hierarchy.

Embedding-based methods (see section 4.1.2) are a further common choice for zero-shot learning (*e.g.*, Frome et al. 2013; Norouzi et al. 2014; Jayathilaka, Mu, and Sattler 2020). However, structural models (see section 4.1.1) can be capable of zero-shot learning as well, as demonstrated in Samplawski et al. (2019) and Zhang, Lertvittayakumjorn, and Guo (2019).

4.2.5. Natural Language Processing

Most of the aforementioned models and methods leverage the knowledge contained in concept hierarchies to increase their performance w.r.t. a visual task. This gap between visual and textual or semantic modalities is not without consequences, as discussed in more detail in section 6.6.1. Collell Talleda (2016) and Deselaers and Ferrari (2011) expand further on the gap between the modalities. However, there are several non-visual machine learning tasks which can benefit from concept hierarchies in a more direct manner, one of which is NLP.

Benkhalifa, Mouradi, and Bouyakhf (2001) tackles a semi-supervised text categorization task on news articles. The authors propose a feature representation that includes knowledge from WordNet (see section 3.3.1.1) in addition to example-specific features. This representation is used in an agglomerative clustering step to generate labels for the unannotated fraction of the training data. Using a concept hierarchy in a zero-shot text categorization task is discussed in Zhang, Lertvittayakumjorn, and Guo (2019).

In Fu et al. (2014), new hyponymy relations between concepts not represented in the concept hierarchy are discovered. These new relations are recognized using a word embedding method (see section 4.1.2). The work focuses on Chinese semantics, which the authors claim can be "easily adapted to other languages." Similar relation discovery tasks are explored in Vendrov et al. (2016), Nickel and Kiela (2017), and Ganea, Becigneul, and Hofmann (2018).

Sentiment classification is a very complex NLP task because it has to consider aspects such as irony and subtext. Hence, reliable annotated training data is rare and not readily available. Dehghani et al. (2017) add noisy training data, *i.e.*, weakly supervised learning (see section 4.2.4.2). The authors leverage a concept hierarchy to estimate the reliability of individual annotations more accurately. Lin, Gao, and Li (2020) considers a stronger semi-supervised scenario (see section 2.1.4.2) with a very small fraction of annotated training data.

5. Semantic Knowledge Integration and Learning from Imprecise Data

In this chapter, we introduce the idea of imprecise data. It is inspired by the concepts of precision and accuracy concerning measurements, which we apply to labels. After a formal definition we first outline some important challenges associated with it. This includes an appropriate mathematical representation and statistical models. Processing of imprecise data by automatic means is then discussed as well.

The following section details the actual learning from imprecise data. We develop a probabilistic hierarchical classifier that is capable of integrating semantic knowledge into the classification process. It is derived from a free probabilistic model by gradually restricting it, introducing assumptions based on semantic knowledge. Additionally, we discuss possible problems with these assumptions that can potentially result in bad performance.

Finally, the task of extrapolating from imprecise data is specified. Even if the training data is imprecise, we expect precise predictions from a model. We describe a modification of the aforementioned classifier that allows it to perform this extrapolation. The model is further developed by adding a self-supervised component which extrapolates even during training.

5.1. Imprecise Data

In this section, we investigate the phenomenon of imprecise data. This phrase also implies the existence of *precise* data. Our use of this terminology is inspired by metrology, the science of measurements, where it originates from. The quality of a numerical measurement, *e.g.*, that of a scale or a thermometer, is typically characterized by *accuracy* and *precision* (*cf.* BIPM et al. 2012).

Accuracy is the "closeness of agreement between a measured quantity value and a true quantity value of a measurand" (ibid., p. 21). This definition differs from the one used in classification (see section 6.1.1.2). There, the frequency of agreement is evaluated because categorical variables are not equipped with a natural notion of "closeness". Precision is the "closeness of agreement between [. . .] measured quantity values obtained by replicate measurements" (ibid., p. 22).

Neither accuracy nor precision can be inferred from a measurement without context. However, there is an agreement to communicate the precision of a measurement in its numerical representation by the number of significant digits. A similar convention can be applied to annotations for classification tasks (*cf.* Chang et al. 2021). For example, a labeler might not be able to distinguish *European Shorthair* and *British Shorthair* consistently. If they were forced to annotate at the level of cat breeds, their *imprecision* would result in alternating annotations which would be partly

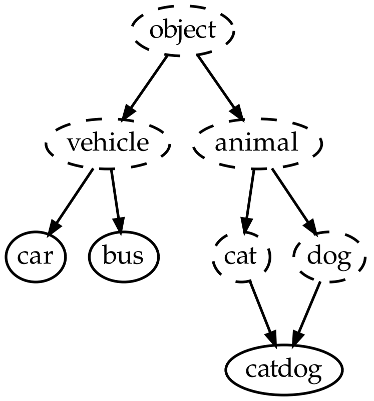


Figure 5.1.: Imprecise (dashed) and precise (solid) concepts in a concept hierarchy.

incorrect. Alternatively, they could be offered to label the image as *Cat* to express their imprecision as part of the label. With this intuition, we can then define:

Definition 5.1 (Precise and imprecise concepts)**.** A concept c is (semantically) *imprecise* if there exist hyponyms of c in the relevant context. If no hyponyms of c exist, then c is a *precise* concept.

Definition 5.2 (Precise and imprecise data)**.** If training data consists at least partly of examples where labels are imprecise concepts, it is called (semantically) *imprecise data*. Otherwise, it is precise data.

Figure 5.1 visualizes this definition in a graph representation of a concept hierarchy (see section 3.2.2). It is important to note that this definition is not applicable without a context (see section 3.1.1). Similar to numerical measurements, there is no absolute or globally valid distinction between precise and imprecise.

Note that there exist types of imprecision other than semantic in machine learning, *e.g.*, in multi-label classification (*cf.* Younes, Abdallah, and Denœux 2010). Furthermore, there is no indication of the actual fraction of imprecise labels in semantically imprecise data except that it is larger than zero. However, for practical purposes, each precise concept in the relevant context should occur at least once if a correct prediction is expected. A task where one or more precise concepts are missing from the training data entirely is called zero-shot learning (see section 4.2.4.3).

In the following, we describe mathematical models of imprecision and tackle the more practical problem of aligning existing datasets and concept hierarchies as well as extracting singular concepts from image descriptions.

5.1.1. Measuring and Modeling Imprecision using Concept Hierarchies

Imbalanced training data, *i.e.*, data where some classes are over- or underrepresented, is a notable challenge in machine learning. For classification tasks, this phenomenon can be modeled in terms of probabilities. If the prior probabilities for each class are known, the imbalance can at least partially be corrected. Probabilities can be considered during validation and the training data can be resampled or weighted to be more representative.

A similar probabilistic model of imprecise data would be helpful, but this data is not as straightforward to model. Concepts are not mutually exclusive and not independent. Hence, assigning a single probability to each concept is not reasonable. Instead, we propose to assign individual probabilities to precise concepts, *i.e.*, leaf nodes only. In addition, we model precision using a parameterized distribution.

We require a numerical measure of precision to be able to model our data in this way. In this section, we explore two approaches to precision modeling based on depth and information content.

5.1.1.1. Depth-Based Models

Partial results of the work presented in this section are published in Brust, Barz, and Denzler (2021b).

A natural approach to modeling semantic precision numerically is the graph distance from the least precise concept, *i.e.*, the root (see section 3.2.2). We also refer to this as *depth* in the hierarchy. In this model, the root, *e.g.*, `entity.n.01`$^{\mathrm{W}}$ is assigned zero precision. A drawback is that not all leaf nodes have the same numerical precision, which means that this type of numerical precision cannot be used to distinguish between precise and imprecise concepts. The advantages of such an approach are the relative simplicity of modeling and sampling and an intuitive visualization of distributions as a histogram. It also relates well to certain concept hierarchies where depths are assigned fixed meanings, *e.g.*, phylum, regnum, etc. in biological taxonomy (see section 3.3.3.1). A method to sample new imprecise data from originally precise data is presented in algorithm 1.

Algorithm 1 Sampling algorithm for imprecise labels from precise concepts.

```
 1: procedure SampleImpreciseLabel($G, c$)          ▷ Graph $G$, precise concept $c$
 2:     $d \leftarrow$ depth sampled from precision distribution.
 3:     if $d_G(\text{root}, c) = d$ then
 4:         return $c$
 5:     else
 6:         $P \leftarrow$ all shortest paths from root to $c$ in $G$
 7:         $p \leftarrow$ path randomly sampled from $P$ with equal probabilities
 8:         $c' \leftarrow$ $d$-th element of $p$
 9:         return $c'$
10:     end if
11: end procedure
```

If G is a DAG, shortest paths are not unique, making the sampling in line 7 necessary. Note that sampling with equal probabilities is technically not correct, as elements of different shortest paths could subsume slightly different leaf nodes. However, the differences are too small to have an effect in real-world applications.

In the following, we give three examples of precision distributions that model specific sources of imprecise data.

Volunteers First, we consider labels provided by volunteers. Volunteers are not paid for their work, but typically have intrinsic motivation to perform to the best of their abilities. Their interest in a specific field also possibly translates to a slightly higher expertise compared to the general population. However, they are rarely formally qualified domain experts, as those can command pay for their abilities. Overall, we expect most labels to be of high to medium precision. Very precise or very imprecise labels are less likely. This expectation is captured by a Poisson distribution.

Web Crawling The second possible source is training data crawled from the web, *e.g.*, image captions from a search result. Imprecision can come from pre-processing errors (see sections 5.1.2.1 and 5.1.2.2). Image captions can also be imprecise on purpose (from a process called *hypernymization*, see section 3.3.2.2). Annotators are not aware that they are in fact annotation and do not typically intend to produce as precise a label as possible. Depending on the context, they may choose to withhold information deliberately in an attempt to "clickbait". Optimizing for search engines (SEO) can also reward overly general image descriptions as they apply to more search terms. We model this source with a geometric distribution, *i.e.*, a power law. This is also described qualitatively in Deng et al. (2014), but not used as part of their evaluation.

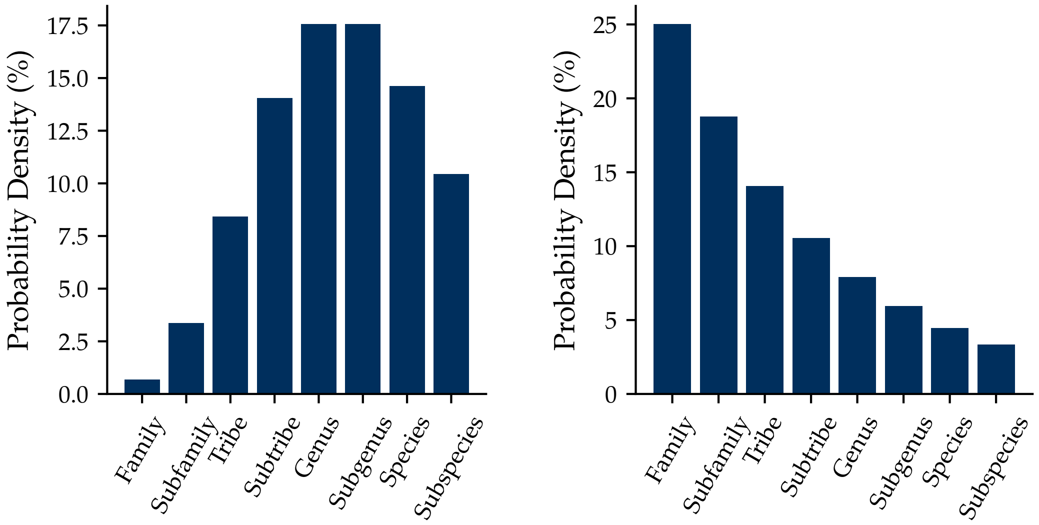
Deng et al. (2014) Instead, Deng et al. describe a simpler model of imprecision, or more precisely, how to generate imprecise data from precise data. We also use this model in our experiments for comparison purposes. Labels are replaced by a direct hypernym with a fixed probability. However, this does not impact precision much in the case of the ImageNet dataset (see section 6.2.1.1) considering the already high depths ≥ 10. Table 6.4 in section 6.4.1.3 quantifies this impact.

Finally, precise data, *e.g.*, a benchmark dataset, is also easily modeled in terms of depth using a Dirac delta distribution. However, if the concept hierarchy has precise concepts at different depths, a distribution over depths cannot correctly describe the data. In any case, sampling can simply be left out when the goal is precise data. We visualize all aforementioned distributions in fig. 5.2. An empirical analysis providing validation on real-world data is presented in section 6.6.3.

5.1.1.2. Information Content

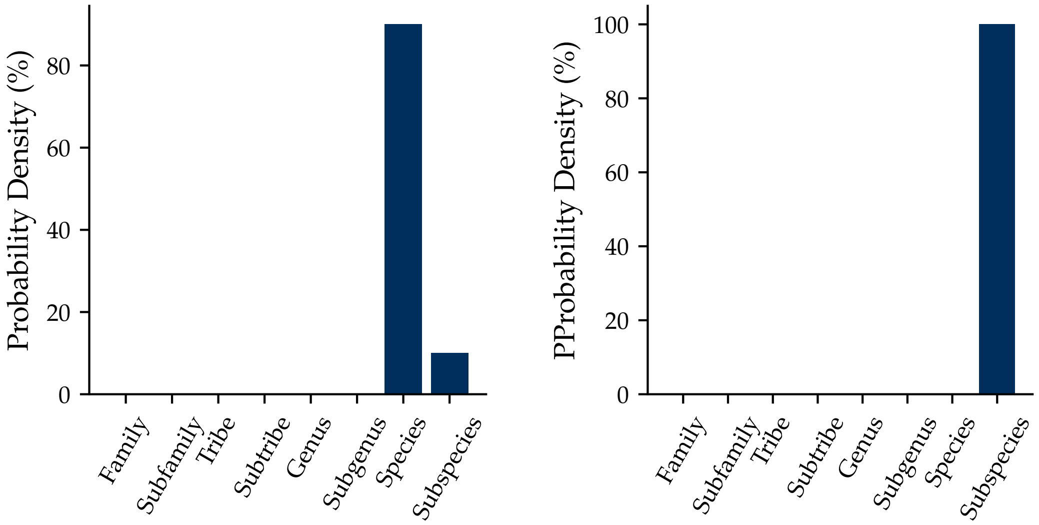
There exist concept hierarchies that match the model of depth as precision (introduced in section 5.1.1.1). For example, the taxonomy in biology (see section 3.3.3.1) assigns a name and meaning to each "level". Other concept hierarchies are unbalanced and extend to different depths in different subhierarchies, *e.g.*, WordNet (*cf.* Miller 1995, also section 3.3.1.1). The resulting discrepancies are not as strong as to invalidate depth-based models, but there are applications (section 5.3.3 for example) that require a more consistent and also continuous measure of semantic precision.

Semantic precision is also known as *concept specificity* in NLP (*cf.* Harispe et al. 2015, 52 sqq.). There, it is measured in terms of information content (IC). IC takes



(a) Poisson distribution (volunteers)

(b) Geometric distribution (web crawling)



(c) Relabeling to hypernyms after Deng et al. (2014)

(d) Benchmark dataset, where all concepts are as precise as possible

Figure 5.2.: Distributions over depth in a concept hierarchy. Numbers are replaced with matching levels of the biological taxonomy to give an application-oriented visualization. Figure taken from Brust, Barz, and Denzler (2021b).

the context of a concept into account and there are two variants of IC which are distinguished by the type of context considered. *Extrinsic* IC views a concept in the context of corpora and is affected by the actual usage of the concept. Conversely, *intrinsic* IC is based on a taxonomy, or concept hierarchy (*cf.* Harispe et al. 2015, p. 55).

There are several competing definitions of intrinsic IC (*cf.* ibid., p. 55). In this work, we use a formula proposed in Zhou, Wang, and Gu (2008) unless specified otherwise. Zhou, Wang, and Gu define their intrinsic IC as:

$$\mathfrak{I}(c) = k\Big(1 - \frac{\log(|E(c)|)}{\log(|\mathcal{C}|)}\Big) + (1-k)\frac{\log\big(d_G(\mathrm{root}, c)\big)}{\log\big(\max_{c' \in \mathcal{C}} d_G(\mathrm{root}, c')\big)}, \tag{5.1}$$

where $E(c)$ is the set of all concepts subsumed by c, including c itself (the extension of c, see section 3.1.1.2). k is a coefficient to balance the model towards either the number of subsumed concepts or the depth in the concept hierarchy. If $k = 0$, the IC is purely depth-based (*cf.* section 5.1.1.1). We use the value $k = 0.6$ as recommended in Harispe et al. (2015, p. 55).

$\mathfrak{I}$ is bounded by $[0, 1]$. The root of the concept hierarchy, *e.g.*, `entity.n.01`$^{\mathrm{W}}$, has an IC of 0. $\mathfrak{I}$'s upper bound of 1 is assigned to the most precise leaf nodes. However, depending on the balance of the hierarchy, not all leaf nodes can reach 1, only if they reside at the maximal distance from the root.

"Measuring" semantic precision by a single number is important for modeling, benchmark and for describing datasets. However, the idea of assigning a number to a concept is not without flaws. Numbers can always be compared and imply a total order. In contrast, not all concepts can be compared in a meaningful way — at least not through the hyponymy relation which can only be a *partial* order. The detailed reasoning is given in section 3.2.1.2.

5.1.2. Mapping Concepts to Datasets

The phrase "integrating domain knowledge" implies that datasets and domain knowledge (in the form of concept hierarchies, see section 3.3) originate from different sources. Specifically, the acquisition of labels is performed without awareness of an existing concept hierarchy. Web crawling is an extreme case where annotators are not even aware of the labeling task itself, much less a restricted set of concepts. In this section, we show what actions can be taken to align a given natural language image description to a concept hierarchy. We focus on WordNet (Miller 1995) as it is the largest lexical database (see section 3.3.1.1) and relevant to many potential applications. In section 6.6.3, we demonstrate an application of the procedures described in the following.

5.1.2.1. Stemming and Lemmatization

Consider an image described as:

> many geese sit in front of a bench overlooking a lake[1]

After removing uninformative *stop words*, the following remains:

> many geese sit front bench overlooking lake

The overall goal is turning this caption into a single-concept label, *i.e.*, `goose.n.01`^{W} or `bench.n.01`^{W}. A first step towards this goal mapping the individual words to lemmata, where a lemma is a word as it would appear in a dictionary (*cf.* Manning, Raghavan, and Schütze 2008, p. 32). Words in practical use are subject to inflection of the original lemma, *e.g.*, be → is. Any mapping algorithm should also consider derivative forms such as poet → poetry, which are different lemmata that share a stem.

Stemming is a simple approach to the mapping problem (*cf.* ibid., p. 32). It consists of a fixed list of rules to remove endings from words and only leave the stem. While the implementation is easy and rules can be derived from a known grammar, it is an incomplete solution. Irregular forms, *e.g.*, geese, are not considered unless part of the list of rules. More importantly, stemming can inadvertently remove derivational affixes, which is dangerous because lemmata differentiate between derivational forms of words and the meaning may change substantially (*cf.* ibid., p. 32).

Alternatively, the more holistic *lemmatization* can be used. It involves detailed morphological analysis and a lexical database to cover any irregular forms. As such, it is the "proper" way to map a word to a lemma (*cf.* ibid., p. 32). The program `MORPHY` is a lemmatizer that ships with WordNet and combines morphological knowledge with access to the vast amount of lexical data available in it. `MORPHY`'s development is detailed in Beckwith and Miller (1990).

5.1.2.2. Lemma Disambiguation

After successful lemmatization, the example in section 5.1.2.1 might look like this:

> many goose sit front bench overlook lake

However, this is still not a usable set of concepts in the form of synsets. One synset can have multiple lemmata that "implement" it. And, crucially, the same lemma can be associated with more than one synset. Deciding which synset is the right for a given lemma without further context or knowledge is an ill-posed task known as *disambiguation*. Even with context, it is a very complicated problem and subject to active research, see *e.g.*, Magnini et al. (2001), Kohli (2021), and AlMousa, Benlamri, and Khoury (2021).

In this thesis, we apply a simple heuristic to map an ambiguous lemma to a synset. We always choose the least precise synset, where precision is defined as the length of the shortest path from `entity.n.01`^{W} to the respective synset in the WordNet hierarchy (see section 5.1.1.1 for details). This selection has the smallest potential for errors. However, it should be noted that when an error occurs, which is less likely than with a different selection, it is also more consequential. Finally, the selected synsets and their definition from WordNet (*cf.* Miller 1995, also section 3.3.1.1) are:

---

[1] From MSCOCO dataset, see `http://cocodataset.org/#explore?id=291980` (last accessed: June 28th, 2021), Lin et al. (2014) and section 3.3.2.2 for more detail.

1. `many.a.01`$^{\mathrm{W}}$: a quantifier that can be used with count nouns and is often preceded by 'as' or 'too' or 'so' or 'that'; amounting to a large but indefinite number
2. `goose.n.03`$^{\mathrm{W}}$: flesh of a goose (domestic or wild)
3. `sit.v.01`$^{\mathrm{W}}$: be seated
4. `front.a.01`$^{\mathrm{W}}$: relating to or located in the front
5. `bench.n.06`$^{\mathrm{W}}$: the reserve players on a team
6. `overlook.v.01`$^{\mathrm{W}}$: look past, fail to notice
7. `lake.n.01`$^{\mathrm{W}}$: a body of (usually fresh) water surrounded by land

5.1.2.3. Further Challenges

Even if the challenges of lemmatization and disambiguation could be overcome, there are still problems that remain. A limitation of the word-to-synset formulation for image captions is the assumption that a single concept can be used to describe the image reasonably well. This assumption is problematic in an of itself (*cf.* Brust and Denzler (2019b) and section 6.6.1), but fundamental to image classification tasks. In this thesis, we always choose the most precise synset, in terms of depth (see section 5.1.1.1) out of a list. For our earlier example, the depths are:

0 for `many.a.01`$^{\mathrm{W}}$, `sit.v.01`$^{\mathrm{W}}$, `front.a.01`$^{\mathrm{W}}$,

1 for `overlook.v.01`$^{\mathrm{W}}$,

4 for `bench.n.06`$^{\mathrm{W}}$, `lake.n.01`$^{\mathrm{W}}$,

8 for `goose.n.03`$^{\mathrm{W}}$.

For general object recognition, we select the most precise noun, *i.e.*, `goose.n.03`$^{\mathrm{W}}$. This result is almost correct, but the large potential for errors is apparent from the examples in sections 5.1.2.1 and 5.1.2.2.

Furthermore, lemmata in WordNet can consist of more than one word, *e.g.*, the lemma "bird of night" of `owl.n.01`$^{\mathrm{W}}$. This means that a one-to-one mapping of words to synset is not necessarily a good choice in the first place. However, if we were to consider many-to-one mappings, we would almost certainly need a method based on machine learning to tackle the combinatorial consequences as enumeration and selection would be intractable otherwise.

Finally, a lemma may in rare cases be associated with synsets of different parts-of-speech, *i.e.*, nouns, verbs, or adjectives. It is impossible to heuristically decide based on measures of precision because they are not comparable between different parts-of-speech. In practice, we limit the algorithms to nouns for object recognition purposes, which can result in spurious associations if the image captions contain verbs or adjectives.

Overall, it is recommended to adapt the label acquisition step of a machine learning project to an existing concept hierarchy in the first place. If that is not possible, the algorithms and heuristics in sections 5.1.2.1 and 5.1.2.2 can be applied to align training data and concept hierarchy after the fact.

5.1.2.4. Exploration of Subhierarchies

For a given dataset, it is unlikely that the whole WordNet hierarchy of nouns is necessary. Instead, a new task-specific context should be built for each dataset by sampling the required parts of WordNet. This also ensures that the leaf nodes of the resulting concept hierarchy are appropriate for the task and not too precise.

Starting from an empty context, an exploration step is performed for each encountered concept. All hypernyms of the respective synset are extracted from WordNet and added to the context together with the respective elements of the hyponymy relation. This process is recursive as the exploration step also applies to the newly found hypernyms. It is repeated until no new hypernyms can be found. Optionally, the concept hierarchy is simplified by removing all concepts with exactly one hyponym and connecting the hypernyms.

When a new concept is added during training, the classification algorithm requires adaption. This can be done without "forgetting" in an incremental learning context (see section 7.1.2).

5.2. Knowledge Integration by Hierarchical Classification

Partial results of the work presented in this section are published in Brust and Denzler (2019a).

This section lays out the design of a probabilistic hierarchical classifier. It is the foundation for CHILLAX (proposed in section 5.3.2) which enables learning from imprecise data. This would not be possible with a conventional CNN or other approaches built on mutually exclusive classes. More importantly, it is an example of integrating domain knowledge (the concept hierarchy) into a machine learning system explicitly and transparently.

Hierarchical classification is not only defined in several ways, but also implemented using a variety of effective approaches (see section 4.1). Embeddings (see section 4.1.2) map labels to points in a low-dimensional vector space, typically such that a distance function in this space matches a semantic distance, *e.g.*, graph distance (see section 3.2.2). They typically only embed precise concepts because it is not clear what the distance between a concept and its hypernym should be. Metric learning (see section 4.1.3) involves learning a distance or similarity metric between two images. The metric can be given explicitly or implicitly, *e.g.*, by specifying margins based on semantic distance. Again it is not clear how a concept and its hypernym could be compared in such a setting.

Instead, we build our classifier in a probabilistic framework. Semantics are modeled explicitly using a concept hierarchy and two simple assumptions (see section 5.2.2).

The inference process has several steps (see section 5.2.3) depending on the concept hierarchy. Hence, when a misclassification occurs, it is easier to hypothesize a cause based on the specific responsible step compared to a one-shot classifier.

We consider two main use cases for our probabilistic hierarchical classifier, which we evaluate individually in sections 6.3 and 6.4:

1. *Integrating domain knowledge*. If a matching concept hierarchy can be obtained, we can improve the accuracy and error quality (*cf.* Bertinetto et al. 2020) of an existing application without requiring additional training data, simply by integrating the concept hierarchy and assumptions. It is crucially important that the assumptions hold (see sections 6.6.1 and 6.6.2)

2. *Learning from semantically imprecise data*. Imprecise data is often available in larger quantity and for lower cost than precise training data. The classifier can correctly interpret the labels and still produce precise predictions with modifications detailed in section 5.3.2.

5.2.1. Internal Representation of Labels

For the use case of integrating domain knowledge, we consider a simple prediction task. The training data is given as $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$, where $\mathcal{X}$ is a set of images and $\mathcal{Y}$ a set of labels. A classification task assumes that $\mathcal{Y} = \mathcal{Y}^P \subseteq \mathcal{C}$ (see section 2.3.2.1), *i.e.*, that each image is associated with a single concept as its label.

Because our primary goal is the improvement of an existing application of classification, we keep these definitions such that the concept hierarchy is not apparent to the application. We also know that the training data only consists of images labeled as concepts in $\mathcal{Y}^P$. However, our probabilistic model is built on a richer label representation, where each individual label is a set of concepts. This *internal label set* $\tilde{\mathcal{Y}}$ formally represents a multi-label classification task (*cf.* section 3.1):

$$\mathcal{P}(\mathcal{Y}^P) \subseteq \tilde{\mathcal{Y}} = \mathcal{P}(\mathcal{C})\,.$$

In the following section, this overly general task is limited using the concept hierarchy and assumptions, resulting in a variant of structured output prediction (*cf.* Shalev-Shwartz and Ben-David 2014, 198 sqq.).

We define a translation from external labels to *internal labels* that encapsulates the internal representation. The internal label is defined as the intension of the external label (see section 3.1.2.2). It is the set of hypernyms of the concept represented by the external label, including the concept itself:

$$\tilde{y}(y) = \{c \in \mathcal{C} : y \leq_{\text{is-a}} c\} = I(y)\,. \tag{5.2}$$

Because the hyponymy relation $\leq_{\text{is-a}}$ is reflexive, there is no need to include y explicitly as part of the equation.

For example, consider a label $y = \texttt{dog}$. We then determine the internal label $\tilde{y} = \{\texttt{object}, \texttt{animal}, \texttt{dog}\}$. We also define a shorthand notation to improve the accessibility of the following sections:

$$\begin{aligned} \tilde{y}_c{}^{+} &\iff c \in \tilde{y} \text{ and} \\ \tilde{y}_c{}^{-} &\iff c \notin \tilde{y}\,, \end{aligned} \tag{5.3}$$

where $\tilde{y} \in \tilde{\mathcal{Y}}$. The expression $\tilde{y}_c{}^+$ is interpreted as c being *given*, or *true*, in $\tilde{y}$. We select this terminology because we initially consider the $\tilde{y}_c{}^+$ for different c independent Bernoulli, *i.e.*, binary random variables. Given a domain point $x \in \mathcal{X}$, a free per-concept hypothesis is then constructed such that:

$$h(x)_c = \mathbb{P}(\tilde{y}_c{}^+|x)\,, \tag{5.4}$$

where $h : \mathcal{X} \to [0,1]^{|\mathcal{C}|}$ (see also section 2.1.2.2). This definition implicitly contains a translation from $\tilde{\mathcal{Y}}$ to $\mathcal{Y}$. The actual hypothesis which maps to a single concept and takes the concept hierarchy into account is given in section 5.2.3.

> In an expression such as $h(x)_c$, we use the concept c as an index. Since the actual order has no practical effect, we omit a mapping from $\mathcal{C}$ to an index $[1, \ldots, |\mathcal{C}|] \cap \mathbb{N}$.

5.2.2. Introducing Assumptions

In this section, we limit the aforementioned free probabilistic model where all concepts are independent. First, we introduce a shorthand notation to allow for a more concise expression of (semantic) dependencies. Consider the *direct* hypernyms $H\uparrow$ of a concept c, which are reachable by the transitive reduction $\leq_{\text{is-a}}{}^-$ of the hyponymy relation $\leq_{\text{is-a}}$. In terms of a DAG, these direct hypernyms are also known as *parent nodes*. We exclude c explicitly because of the reflexivity of $\leq_{\text{is-a}}$:

$$H\uparrow(c) = \{c' : c \leq_{\text{is-a}}{}^- c'\} \setminus \{c\}\,.$$

Furthermore, we define a shorthand for the situation where *any* or *no* hypernym of c is given. It visually matches eq. (5.3), but generalizes from individual to sets of internal labels:

$$\begin{aligned} \tilde{y}_{H\uparrow c}{}^+ &= \bigvee_{c' \in H\uparrow(c)} \tilde{y}_{c'}{}^+ \\ \tilde{y}_{H\uparrow c}{}^- &= \bigwedge_{c' \in H\uparrow(c)} \tilde{y}_{c'}{}^- \end{aligned}$$

We use $\tilde{y}_c{}^+|x$ as a random event which has a unique probability. While $\tilde{y}_{H\uparrow c}{}^+|x$ is not a random event with an own variable, its probability can be calculated:

$$\mathbb{P}(\tilde{y}_{H\uparrow c}{}^+|x) = 1 - \prod_{c' \in H\uparrow(c)} \left(1 - \mathbb{P}(\tilde{y}_{c'}{}^+|x)\right)\,. \tag{5.5}$$

5.2.2.1. Assumptions

We now introduce our assumptions:

1. *Closed world*. We assume that only concepts in $\mathcal{C}$ will ever be encountered. Since the root of the concept hierarchy, *e.g.*, `entity.n.01`$^\text{W}$ in WordNet (see section 3.3.1.1), is a hypernym to all other concepts, its probability is one:

$$\mathbb{P}(\tilde{y}_{\text{root}}{}^+) = 1\,. \tag{5.6}$$

2. *Subsumer required*. Concepts can only occur together with their hypernyms. We formulate this assumption in terms of probabilities:

$$\mathbb{P}({\tilde{y}_c}^{+}|x) \leq \mathbb{P}({\tilde{y}_{H\uparrow c}}^{+}|x) \tag{5.7}$$

$$\implies \mathbb{P}({\tilde{y}_c}^{+}|{\tilde{y}_{H\uparrow c}}^{-}, x) = 0\,. \tag{5.8}$$

   While eq. (5.7) only refers to the *direct* hypernyms of a concept, the rule does apply recursively. Effectively, a concept can only be present with all of its hypernyms.

These assumptions are the full extent of domain knowledge that is integrated into the probabilistic model. There are other possible assumptions, most importantly around exclusion (*cf.* Deng et al. 2014). For example, a *cat* excludes a *dog* which our model does not state explicitly. Instead, the classifier has to learn this relation. However, this is a consequence of allowing DAGs as concept hierarchies. While exclusion is easily modeled correctly when the hierarchy is a tree, cycles in the graph make this assumption unreliable. Consider for example the existence of a *catdog*, which would prohibit the mutual exclusion of *cat* and *dog* from a semantic perspective. Modeling exclusion would also further complicate the inference step if probabilities are to be computed accurately (*cf.* Chen et al. 2015a).

5.2.2.2. Modeling Dependencies

We begin building our probabilistic model $\mathbb{P}({\tilde{y}_c}^{+}|x)$ by decomposition. The first step is introducing the probabilities of complementary events $\mathbb{P}({\tilde{y}_{H\uparrow c}}^{+}|x)$ and $\mathbb{P}({\tilde{y}_{H\uparrow c}}^{-}|x)$:

$$\mathbb{P}({\tilde{y}_c}^{+}|x) = \mathbb{P}({\tilde{y}_c}^{+}, {\tilde{y}_{H\uparrow c}}^{+}|x) + \mathbb{P}({\tilde{y}_c}^{+}, {\tilde{y}_{H\uparrow c}}^{-}|x)\,,$$

which treats $\mathbb{P}({\tilde{y}_c}^{+}|x)$ as a marginal distribution whose marginalization is reverted. We then condition the joint probabilities on $\mathbb{P}({\tilde{y}_{H\uparrow c}}^{+}|x)$ and $\mathbb{P}({\tilde{y}_{H\uparrow c}}^{-}|x)$, respectively:

$$\mathbb{P}({\tilde{y}_c}^{+}|x) = \mathbb{P}({\tilde{y}_c}^{+}|{\tilde{y}_{H\uparrow c}}^{+}, x)\mathbb{P}({\tilde{y}_{H\uparrow c}}^{+}|x) + \mathbb{P}({\tilde{y}_c}^{+}|{\tilde{y}_{H\uparrow c}}^{-}, x)\mathbb{P}({\tilde{y}_{H\uparrow c}}^{-}|x)\,.$$

This representation allows us to apply our second assumption (eq. (5.8)) such that the second summand can be removed. The final probabilistic model for inter-concept dependencies is:

$$\mathbb{P}({\tilde{y}_c}^{+}|x) = \mathbb{P}({\tilde{y}_c}^{+}|{\tilde{y}_{H\uparrow c}}^{+}, x)\mathbb{P}({\tilde{y}_{H\uparrow c}}^{+}|x)\,, \tag{5.9}$$

and, to end the recursion we apply the first assumption (eq. (5.6)):

$$\mathbb{P}({\tilde{y}_{\text{root}}}^{+}) = 1\,.$$

5.2.3. Hypothesis

The equations in section 5.2.2.2 model the dependencies between concepts. Given all relevant $\mathbb{P}({\tilde{y}_c}^{+}|{\tilde{y}_{H\uparrow c}}^{+}, x)$, we can calculate the probabilities $\mathbb{P}({\tilde{y}_c}^{+}|x)$ to predict the concept of an image x. For a complete hypothesis there are two more requirements. First, we need to account for the label set $\mathcal{Y}^P$, which may only be a subset of $\mathcal{C}$. Second, the events ${\tilde{y}_c}^{+}$ pertain to the presence of concepts in a set, an internal label

$\tilde{y} \in \tilde{\mathcal{Y}}$. However, our predictions should be in $\mathcal{Y} = \mathcal{Y}^P$. While we give a translation step from $\mathcal{Y}$ to $\tilde{\mathcal{Y}}$ in eq. (5.2), the inverse mapping remains unspecified.

In the following, we develop a hypothesis which implements the equations from section 5.2.2.2 while taking the aforementioned requirements into account. We begin with the outer hypothesis, which maximizes over all concepts in $\mathcal{Y}^P$ to address the first and second requirements simultaneously:

$$h(x) = \arg\max_{c \in \mathcal{Y}^P} h(x)_c \,.$$

We insert eq. (5.4) to obtain:

$$h(x) = \arg\max_{c \in \mathcal{Y}^P} \mathbb{P}(\tilde{y}_c{}^+ | x)\,,$$

and apply the dependencies in eq. (5.9):

$$h(x) = \arg\max_{c \in \mathcal{Y}^P} \mathbb{P}(\tilde{y}_c{}^+ | \tilde{y}_{H\uparrow c}{}^+, x) \;\; \mathbb{P}(\tilde{y}_{H\uparrow c}{}^+ | x)\,.$$

This recurses until the root of the concept hierarchy is reached, and we obtain

$$h(x) = \arg\max_{c \in \mathcal{Y}^P} \mathbb{P}(\tilde{y}_c{}^+ | \tilde{y}_{H\uparrow c}{}^+, x) \mathbb{P}(\tilde{y}_{H\uparrow c}{}^+ | \tilde{y}_{H\uparrow H\uparrow c}{}^+, x) \ldots \mathbb{P}(\tilde{y}_{\text{root}} | x)\,.$$

With eq. (5.6) we can simplify slightly to:

$$h(x) = \arg\max_{c \in \mathcal{Y}^P} \mathbb{P}(\tilde{y}_c{}^+ | \tilde{y}_{H\uparrow c}{}^+, x) \mathbb{P}(\tilde{y}_{H\uparrow c}{}^+ | \tilde{y}_{H\uparrow H\uparrow c}{}^+, x) \ldots\,.$$

Finally, we express the hypothesis using subhypotheses h':

$$h(x) = \arg\max_{c \in \mathcal{Y}^P} h'(x)_c \cdot h'(x)_{H\uparrow c} \cdot \ldots\,, \tag{5.10}$$

where

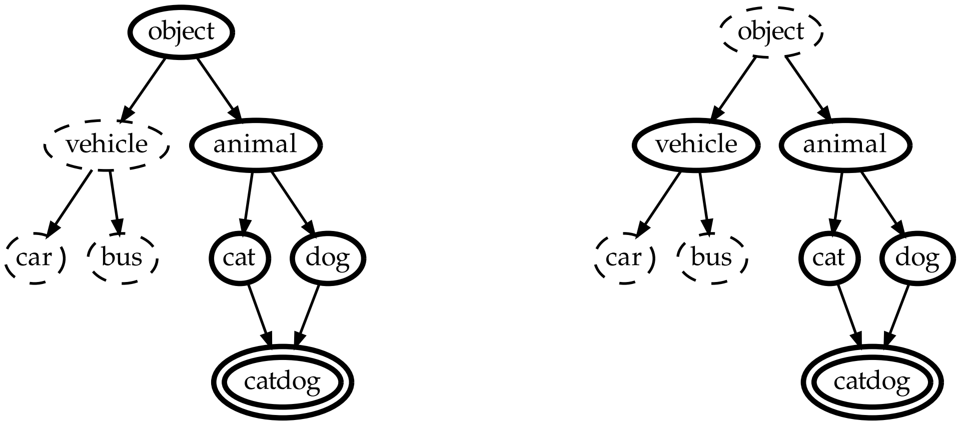
$$h'(x)_c = \mathbb{P}(\tilde{y}_c{}^+ | \tilde{y}_{H\uparrow c}{}^+, x)\,,$$

and

$$h'(x)_{\text{root}} = 1\,. \tag{5.11}$$

With these additions, the model is complete. However, there are considerations w.r.t. $\mathcal{Y}^P$. Namely, if $\mathcal{Y}^P = \mathcal{C}$, the hypothesis is the root regardless of which x is presented. Precise concepts are *almost never* predicted because their probabilities are always lower than or equal to that of their hypernyms due to eq. (5.7). Thus, for the use case of integrating domain knowledge into an existing classification task, $\mathcal{Y}^P$ should only contain precise concepts that are mutually exclusive. This restriction is required by the application in any case, so it is not a relevant limitation of the method.

The hypothesis $h(x)$ can only be expressed in the simple chain form of eq. (5.10) if all concepts along the chain have only one hypernym. If there is more than one hypernym, the expression is more complicated. The correct value is obtained by applying eq. (5.5). Note that this leads to repeating terms in the equation which offer an opportunity for dynamic programming (*cf.*, *e.g.*, Bellman 1957).



(a) Concepts in internal label representation eq. (5.12).

(b) Concepts that fulfill the right-hand side of eq. (5.13).

Figure 5.3.: Visualization of label encoding and loss function mask for the precise label `catdog`. Dashed nodes represent 0 and bold nodes 1. Figure taken from Brust and Denzler (2019a).

5.2.3.1. Imprecise Predictions

An application might also allow imprecise predictions or even require them (*e.g.* Deng et al. 2012; Wu et al. 2020). However, there is more than one possible interpretation of such a prediction, depending on the concept hierarchy and the application scenario. Expressing uncertainty is a useful consideration, where the precision of a prediction is lowered if the possibility of a misclassification becomes too high. We explore this angle in more detail in section 5.3.3.

Alternatively, it could also point to an unexplored concept in an open-world scenario. It does not necessarily violate eq. (5.6). If, *e.g.*, a new species of `animal` is observed, it is still an `object` and an `animal`, but not part of the concept hierarchy. For this scenario, we define an alternative hypothesis which includes the probability $\mathbb{P}(\tilde{y}_{H\downarrow c}{}^{-} | \tilde{y}_c^{+}, x)$ that none of the hyponyms of a concept are true:

$$h(x)_c = \mathbb{P}(\tilde{y}_{H\downarrow c}{}^{-} | \tilde{y}_c^{+}, x) \;\; \mathbb{P}(\tilde{y}_c{}^{+} | \tilde{y}_{H\uparrow c}{}^{+}, x) \;\; \mathbb{P}(\tilde{y}_{H\uparrow c}{}^{+} | x) \, ,$$

which similarly to eq. (5.10) can be expressed as:

$$h(x) = \operatorname*{arg\,max}_{c \in \mathcal{Y}^P} h'(x)_{H\downarrow c} \cdot h'(x)_c \cdot h'(x)_{H\uparrow c} \cdot \ldots$$

This hypothesis formulation "naturally" predicts precise concepts even if $\mathcal{Y}^P$ is not limited to them. It will output imprecise predictions to express missing concepts in the hierarchy, effectively performing novelty detection (see section 7.1.1).

5.2.4. Deep Learning Implementation

In the following, we describe an implementation of the probabilistic model in section 5.2.3 using a deep learning method. The central idea is using the last layer of a

CNN to learn the conditional probabilities $\mathbb{P}(\tilde{y}_c{}^+ | \tilde{y}_{H\uparrow c}{}^+, x)$ such that:

$$h_L(x)_c = h'(x)_c = \mathbb{P}(\tilde{y}_c{}^+ | \tilde{y}_{H\uparrow c}{}^+, x)\,.$$

There are three concerns regarding this implementation. First, we have to define a numerical label representation encoding the internal labels. Second, the last layer of the CNN requires appropriate dimensions and a matching activation function (see section 2.4.1.3). And, finally, a loss function that accounts for the conditional nature of the h' is needed.

5.2.4.1. Label Representation

We use a many-hot encoding representing an internal label's constituent concepts. An internal label is translated from an external label using eq. (5.2). The encoding $e : \mathcal{Y}^P \rightarrow \{0,1\}^{|\mathcal{C}|}$ is then defined as:

$$e(y)_c = \begin{cases} 1 & \text{if } c \in \tilde{y}(y) \\ 0 & \text{if } c \notin \tilde{y}(y) \end{cases}. \tag{5.12}$$

Figure 5.3a visualizes the encoding.

The last layer h_L should output a $|\mathcal{C}|$-dimensional vector similar to e. A sigmoid activation function (see section 2.4.1.3) ensures that each element can be interpreted as a probability.

5.2.4.2. Loss Function

Finally, we define a loss function with two main components. A binary cross-entropy loss function $\mathcal{L}_X$ (see section 2.4.1.4) is used to compare the probabilities. It is multiplied with a mask that represents the condition state of the conditional probabilities in h'. We give a visual example of this mask in fig. 5.3b. The whole loss function is defined as:

$$\mathcal{L}(h_L, x, y) = \sum_{c \in \mathcal{C}} \mathcal{L}_X(e(y)_c, h_L(x)_c) \cdot \begin{cases} 1 & \text{if } \exists c' \in \mathcal{C} : c \leq_{\text{is-a}}{}^{-} c' \wedge y \leq_{\text{is-a}}{}^{-} c' \\ 0 & \text{else} \end{cases}. \tag{5.13}$$

This formulation has the side effect that the weights of the last layer representing the root of the concept hierarchy are not changed in almost all cases (except for hierarchies where the highest depth is 1). Hence, they should be initialized such that the output is a constant 1. We discuss the implications for imprecise labels in section 5.3.2.1 and explore a special case in section 6.5.1.3.

5.2.4.3. Implementation Details

Note that both eq. (5.12) and eq. (5.13) are operations on external labels from $\mathcal{Y}^P$. Internal labels can be completely encapsulated in an implementation and not visible to the application during both training and prediction.

However, while the training process is fully implemented using eq. (5.12) and eq. (5.13) in a pre-existing CNN, predictions require further consideration. An application expects a hypothesis h while the network only provides the output h_L of the last layer. For predictions, eq. (5.10) has to be implemented separately.

5.3. Extrapolating Imprecise Data

Partial results of the work presented in this section are published in Brust, Barz, and Denzler (2021b).

In this section, we develop ideas and methods to learn from imprecise data. A more detailed description of imprecise data is offered in section 5.1 and a number of potential sources are listed in section 5.1.1.1.

First, we formally define the task. We then explore the concept of extrapolation and its role in generalization. Afterwards, we provide a detailed description of our method CHILLAX which implements learning from imprecise data. CHILLAX is finally augmented with a self-supervised learning strategy to further improve sample efficiency.

5.3.1. Problem Formulation

In the previous section 5.2, we first consider different label sets. The set $\mathcal{Y}^P$ contains labels found in the training data and used by an application. With the integrated external knowledge of $\mathcal{C}$ (the set of concepts) and the hyponymy relation $\leq_{\text{is-a}}$ we construct an internal label set $\tilde{\mathcal{Y}}$. However, $\tilde{\mathcal{Y}}$ has no effect on the training data: both the training and the validation set are elements of $\mathcal{X} \times \mathcal{Y}^P$. These pairs (x, y) are then used to generate a hypothesis $h : \mathcal{X} \rightarrow \mathcal{Y}^P$.

For this section, we look at a more complex task, where we differentiate between label sets for training and for validation and prediction. Given a set of concepts $\mathcal{C}$, the training label set $\mathcal{Y}^T$ should be a (non-strict) subset of $\mathcal{C}$. At the same time, the validation and prediction label set $\mathcal{Y}^P$ should be a strict subset of $\mathcal{Y}^T$, *i.e.*, there should be concepts in the training data that are not in the validation set and are not expected to be predicted. Formally, we require $\mathcal{Y}^P \subset \mathcal{Y}^T \subseteq \mathcal{C}$. Hence, the training examples are pairs in $\mathcal{X} \times \mathcal{Y}^T$. However, the hypothesis is still a function $h : \mathcal{X} \rightarrow \mathcal{Y}^P$ and the validation examples are pairs in $\mathcal{X} \times \mathcal{Y}^P$. This problem setting is also referred to as weakly supervised learning (see sections 2.1.4.3 and 4.2.4.2). We visualize the different label sets in fig. 5.4.

If $\mathcal{Y}^P$ contains only *precise* concepts of $\mathcal{C}$, while $\mathcal{Y}^T$ is additionally comprised of hypernyms of $\mathcal{Y}^P$, we name the task *learning from imprecise data*. Often, we have $\mathcal{Y}^T = \mathcal{C}$.

5.3.1.1. Generalization

When learning from imprecise data, the generalization effort is considerably higher than for a conventional prediction task. It requires not only generalization from a set of representative training data to unseen examples. There is an additional level of generalization, from a small amount of precise labels and many imprecise labels to consistently precise predictions.

We term this second instance of generalization *annotation extrapolation* from imprecise to precise concepts. This is not a perfect term because it could be understood to imply generalization to novel, previously unseen concepts, a task otherwise known as zero-shot learning (see section 4.2.4.3). An alternative image for this type of generalization is a *projection*, where an imprecise concept casts a shadow in the light cone

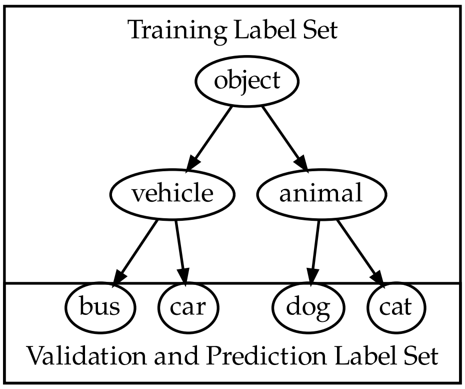


Figure 5.4.: Examples of training label set $\mathcal{Y}^T$ and validation and prediction label set $\mathcal{Y}^P$.

of the concept hierarchy. This shadow is then visible on the "floor" — the precise concepts. Dhall et al. (2020) and Ganea, Becigneul, and Hofmann (2018) use similar imagery to develop an embedding-based method (see section 4.1.2).

5.3.1.2. Practical Considerations

Learning from imprecise data has important implications for evaluation and validation. We must be cautious to avoid situations in which imprecise labels ($\mathcal{Y}^T$) and precise labels ($\mathcal{Y}^P$) are compared. The qualitative difference between training data and validation data makes advanced approaches like cross-validation (see section 2.2.2.2) impossible, because a different split of the dataset might violate the $\mathcal{Y}^P \subset \mathcal{Y}^T$ condition.

Evaluating the accuracy (see section 6.1.1.2) of predictions on a pre-defined validation set is still possible and is sufficient for many applications. However, it is also common practice to observe differences between training and validation accuracy in order to estimate the impact or presence of overfitting (see section 2.2). This strategy is impossible when learning from imprecise data because the accuracy of predictions on the training data (re-classification) cannot be calculated correctly. And while a loss function can be estimated on both training and validation sets, it is also not comparable if the sets differ in precision. Hierarchical measures that take a concept hierarchy into account are a viable alternative. For example, hierarchical precision and recall (see section 6.1.2) are metrics that compare set representations of labels similar to internal labels from section 5.2.1. However, while they allow for comparisons, they do not resolve the qualitative differences between training and validation data.

A further consideration for a practical application is the communication with annotators. Because the acceptance of imprecise data for training already constitutes a significant trade-off, the instructions to the annotators must clearly relate the desired interpretation of imprecise labels. The competing interpretations are discussed in more detail in section 5.3.2.1.

5.3.2. From Hierarchical Classifier to CHILLAX

Partial results of the work presented in this section are published in Brust, Barz, and Denzler (2021b).

This section describes our method for learning from imprecise data, which we call CHILLAX. The name is short for **c**lass **h**ierarchies for **i**mprecise **l**abel **l**earning and **a**nnotation e**x**trapolation.

It is based on the probabilistic hierarchical classifier, which is proposed in section 5.2 and published in Brust and Denzler (2019a). We modify both the input and the output components of the method. The input is adapted to the correct interpretation of imprecise labels (see section 5.3.2.1). We restrict the output to precise concepts only to build a hypothesis $h : \mathcal{X} \rightarrow \mathcal{Y}^P$. These modifications are minor overall, as most of the "heavy lifting" w.r.t. concept hierarchies is done by the hierarchical classifier already.

5.3.2.1. Interpretation of Labels

The hierarchical classifier (section 5.2) expects solely precise labels. If there is an imprecise label, the interpretation is that it is imprecise only because none of its hyponyms apply, as a consequence of an incomplete concept hierarchy (see section 5.2.3.1). The label is viewed to be as precise as possible given the specific example and concept hierarchy.

With CHILLAX, we interpret imprecise data differently. An imprecise label means that the correct precise hyponym is probably a part of the concept hierarchy, but unknown to the annotator. To represent this new interpretation, we modify the binary mask in the loss function in eq. (5.13). The new loss function is defined as:

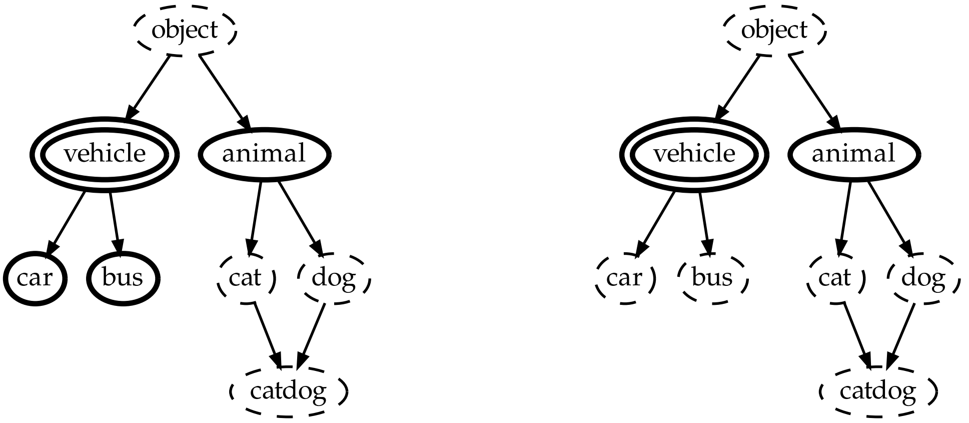
$$\mathcal{L}(h_L, x, y) = \sum_{c \in \mathcal{C}} \mathcal{L}_X(e(y)_c, h_L(x)_c) \cdot \begin{cases} 1 & \text{if } \exists c' \in \mathcal{C} \setminus y : c \leq_{\text{is-a}^-} c' \wedge y \leq_{\text{is-a}^-} c' \\ 0 & \text{else} \end{cases} . \tag{5.14}$$

In fig. 5.5, we visualize the differences between eq. (5.13) and eq. (5.14).

Excluding the ground truth concept y from the list of possible concepts c' ensures that weights representing the hyponyms of y are not changed. As a side effect of this modification, it is not necessary to change the label encoding in eq. (5.12). Because the relevant parts of this representation would be ignored by eq. (5.14), we apply eq. (5.12) directly to CHILLAX.

5.3.2.2. Restricting Inference

The inference process of the hierarchical classifier (section 5.2) as described in section 5.2.3.1 is capable of generating imprecise predictions. However, we require only precise concepts as predictions. Hence, we can apply the more simple process in eq. (5.10) without including the hyponyms of each candidate as a factor. Furthermore, because of the different loss function (eq. (5.14)), using the process in section 5.2.3.1 would yield false predictions as the corresponding weights are never learned correctly.



(a) Probabilistic hierarchical classifier loss function mask (eq. (5.13)).

(b) CHILLAX modified loss function mask (eq. (5.14)).

Figure 5.5.: Visualization of loss function masks for the imprecise label `vehicle`. Dashed nodes represent 0 and bold nodes 1. Figure taken from Brust, Barz, and Denzler (2021b).

If the training data consists only of precise examples, *i.e.*, $\mathcal{Y}^T = \mathcal{Y}^P$, CHILLAX behaves identically to the unmodified hierarchical classifier in section 5.2.

5.3.3. Self-Supervised Approach

Partial results of the work presented in this section are published as a pre-print in Brust, Barz, and Denzler (2022).

While CHILLAX (see section 5.3.2) fulfills the requirements for learning from imprecise data, it is not optimal. The loss function eq. (5.14) and its CHILLAX counterpart eq. (5.13), together with the closed-world assumption in eq. (5.6), have a negative effect on imprecisely labeled examples. Training data examples that are annotated as the minimally precise concept (the root of the concept hierarchy) always produce a loss $\mathcal{L} = 0$.

Still, we observe that CHILLAX can learn from imprecise data even in extreme cases, where only a small fraction of the training data is precise, and nevertheless make predictions with a relatively much larger accuracy (see section 6.4). This generalization capability is called extrapolation (see section 5.3.1.1). In the following, we utilize this capability not only for predictions, but already during training. The strategy is designed to tackle the problem of effectively unlabeled examples.

Our main scheme is to replace the very or maximally imprecise labels of the training data with more precise *pseudo-labels*. The setup is an example of self-supervised learning (see sections 2.1.4.3 and 4.2.4.2). It is termed *self*-supervised because we compute the pseudo-labels using predictions of the same classifier that learns from the pseudo-labels.

In a conventional classification task, we consider the predicted class scores, or probabilities. For example, a hypothesis generates the scores:

$$(0.1, 0.3, 0.4, 0.05, 0.15)\,.$$

If this distribution meets our requirements, *e.g.*, with high enough scores, we generate a pseudo-label by extrapolating the probabilities (*cf.* Wang et al. 2017b) to:

$$(0, 0, 1, 0, 0)\,.$$

The new pseudo-label is then used as ground truth for training. Pseudo-labels are recomputed continually to reflect changed hypotheses. Note that if we skip the extrapolation step and attempt to learn the predicted probabilities directly, the corresponding loss function would evaluate to zero. Hence, the extrapolation is necessary. In the following, we describe methods that build on this concept, but advance it by increasing not only the individual probabilities, but the precision of the label.

Formally, we are looking for a mapping from an imprecise example (x, y) to a precise example $(x, \hat{y})$. The label $y \in \mathcal{Y}^T$ can be an imprecise concept, and we call y the extrapolation *source*. Consequently, the pseudo-label $\hat{y} \in \mathcal{Y}^T$ is the extrapolation *target*. We do not change the domain point x with the mapping. The following sections contain several heuristics that implement such a mapping. A common component is a criterion to first build a set of candidate concepts $\hat{\mathcal{Y}}$ and elect the most informative or confident candidate concept as a pseudo-label.

5.3.3.1. Known and Unknown Probabilities

While there are many possibilities for a mapping $(x, y) \mapsto (x, \hat{y})$, we can exclude a subset by imposing one obvious restriction on the pseudo-label $\hat{y}$. It should never "disagree" with the extrapolation source y. Formally, the condition $\hat{y} \leq_{\text{is-a}} y$ should *always* hold true.

We can use this restriction to our advantage because y contains vital information, unless it is the root. Instead of computing pseudo-labels purely from predictions based on x, we can replace individual subhypotheses $h'(x)_c$ in eq. (5.10) with 0 or 1 wherever they can be determined from the source concept y. Figure 5.6 visualizes the contributions of the ground truth source and the subhypotheses that make up a final prediction.

5.3.3.2. Confidence-Based Method

Given a pair (x, y), our goal is to extrapolate a source concept y to a more precise concept $\hat{y}$. After applying the replacements as in section 5.3.3.1, we examine the individual hypotheses $h(x)_c$ as described in eq. (5.4) for all concepts in $\mathcal{C}$. We learn in section 5.2.3 that comparing $h(x)_c$ across the whole set of concepts $\mathcal{C}$ always leaves the least precise concept (the root) as the most probable. Moreover, after replacing individual subhypotheses $h'(x)_c$ with the ground truth y where applicable, the maximum becomes ambiguous unless y is the root, which further complicates the situation. Hence, in the following, we describe heuristics that do not rely on extreme values.

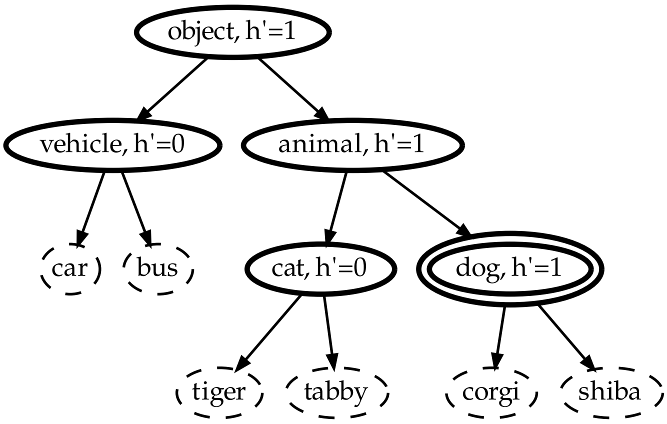


Figure 5.6.: Subhypotheses $h'(x)_c$ for label `dog`. Bold nodes represent known, dashed nodes unknown subhypotheses.

Method: Fixed Threshold Instead of considering the most probable concept, we interpret the predicted probabilities $h(x)_c$ as confidence scores (*cf.* Guo et al. 2017) and define a minimum confidence threshold θ to determine a reliable prediction. A similar process is proposed in Wang et al. (2017b). The following definition describes a set of candidate concepts $\hat{\mathcal{Y}}$ that are considered reasonable as a pseudo-label:

$$\hat{\mathcal{Y}}(x) = \{c \in \mathcal{C} : h(x)_c \geq \theta\}\,.$$

This equation explicitly fulfills the condition $\hat{y} \leq_{\text{is-a}} y$ (see section 5.3.3.1) by only considering hyponyms of y. The threshold θ determines a reliable confidence value. It should be > 0.50 to exclude the exact 0.50 probability given by zero-initialized untrained elements of the last layer. The final pseudo-label is the candidate concept with the highest IC $\mathfrak{I}$ (see section 5.1.1.2):

$$\hat{y} = \arg\max_{c \in \hat{\mathcal{Y}}} \mathfrak{I}(c)\,.$$

If multiple concepts are tied for highest IC, the pseudo-label is the tied concept with the highest confidence $h(x)_c$.

We add Gaussian noise with $\sigma = 0.00$ to all $h(x)_c$ independently. This strategy prevents a non-random selection of concepts that are tied for both IC and confidence. A tie often happens at the beginning of training where many $h(x)_c$ are equal, *i.e.*, 0.50.

5.3.3.3. Hierarchy-Informed Methods

The "fixed threshold" method described in the previous section 5.3.3.2 has one main drawback. It suffers from a potential feedback loop. In the introduction to section 5.3.3, we discuss the extrapolation of confidence values, *e.g.*, from 0.40 to 1.

After changing the neural network's weights to minimize the loss function w.r.t. this pseudo-label, the exact same domain point x is predicted with different confidence values. Specifically, the maximum will be higher, *e.g.*, 0.60 instead of the previous 0.40.

This localized increase in confidence has two consequences. First, the average confidence of the highest-scoring concept increases over time. A fixed threshold θ that is optimal for one time step has to be corrected for subsequent iterations. Second, consider a situation where the actual correct concept in the previous example is one with a confidence score of 0.30, while the concept assigned 0.40 is a misclassification. After learning the incorrect pseudo-label, the error is *amplified* and the difference in scores increases, in turn making it less likely that the correct concept is selected as a pseudo-label later. This feedback loop is a fundamental problem of self-supervised learning based on extrapolation of confidence scores.

While there is no direct solution for the second point, we can avoid the first consequence by developing a method that relies less on interpreting absolute confidence values correctly. Instead, we use the concept hierarchy to guide the initial selection of candidate concepts.

Method: Leaf Node This method is the most direct application of the pseudo-label approach to self-supervised learning, which is to use (confident) predictions as training data. For the overall task, we expect predictions to be only precise concepts. Hence, the pseudo-labels should be precise as well. We define the candidate set as the set of precise concepts, *i.e.*, leaf nodes of the concept hierarchy graph, that are also hyponyms of the extrapolation source:

$$\hat{\mathcal{Y}}(x,y) = \{c \in \mathcal{C} : c \ \leq_{\text{is-a}} y \land \nexists c' \in \mathcal{C} : c' \ \leq_{\text{is-a}} c\} .$$

The final selection of a pseudo-label from this set is identical to the process described in section 5.3.3.2.

Method: d^* Steps For this method, we initially consider the set of *all* hyponyms of the extrapolation source (its extension, see section 3.1.1.2) instead of only the most precise to decrease the potential for misclassifications. More precise concepts are more likely to be misclassified for two reasons. First, there are simply more concepts to choose from at more precise levels in a concept hierarchy. Second, there are more individual decisions involved, or to be precise, a multiplication of more individual probabilities (subhypotheses). We further assume that there is a "sweet spot" precision level relative to the extrapolation source's precision where an optimal trade-off can be achieved. To implement this idea, we model precision numerically using depth (see section 5.1.1.1). We define an optimal distance d^* from the extrapolation source y in the concept hierarchy graph G to build a set of candidates:

$$\hat{\mathcal{Y}}(x,y) = \{c \in \mathcal{C} : c \ \leq_{\text{is-a}} y \land d_G(y,c) = d^*\} .$$

As evident from the formula, we only allow concepts where the distance is exactly d^*. However, it is possible that there exist no concepts at the given distance. Consequently, we also allow precise concepts at a *lower* distance. Note that the distance is directional in all cases, *i.e.*, we only tolerate more precise concepts, not less precise ones. This is enforced by the condition $c \leq_{\text{is-a}} y$.

We recommend an optional low threshold, *e.g.*, 0.55, to further narrow down the set of candidates. The remaining process is equal to section 5.3.3.2.

Method: IC Range We discuss the general limitations of modeling concept precision using graph distances or depth in section 5.1.1.2, which also affect the previously presented method "d^* steps". There is a further drawback for practical use, namely, that there are very few different settings because graph distance is a coarse measurement. For a more fine-grained approach, we instead consider the expected increase in IC (see section 5.1.1.2) between the extrapolation source and candidate pseudo-label. Because it is unlikely that a match for an exact number exists, we define an allowed range of IC gain $[\delta\mathfrak{I}_{\min}, \delta\mathfrak{I}_{\max}]$. This range is used to compute the set of candidates as follows:

$$\hat{\mathcal{Y}}(x, y) = \{c \in \mathcal{C} : c \leq_{\text{is-a}} y \wedge \delta\mathfrak{I}_{\min} \leq (\mathfrak{I}(c) - \mathfrak{I}(y)) \leq \delta\mathfrak{I}_{\max}\}\,.$$

Depending on the range, this method suffers from the same problem as "d^* steps", namely that there might not be any candidate concepts. We apply the same solution, and also recommend the same optional threshold and perform the final selection as in section 5.3.3.2.

All of the above definitions contain the condition $c \in \mathcal{C} : c \leq_{\text{is-a}} y$. This condition is technically redundant, because predicted probabilities are always considered for selection. The probabilities are 0 for concepts that are not hyponyms of the extrapolation source. We include the extraneous condition for a better intuitive understanding of the individual methods.

5.3.3.4. Adaptive Method

We propose the methods "d^* steps" and "IC range" as alternatives to a fixed threshold based purely on structural criteria, where structural pertains to the concept hierarchy. However, these methods may not apply equally to all concepts in question, or across a large precision range. In the following, we aim to relax the hard restrictions of the structural methods, but still reach a fixed expected IC gain.

Method: Adaptive Threshold We define a threshold θ in the same way as section 5.3.3.2, but allow its value to change throughout training. The goal is to approach an expected IC gain $\delta\mathfrak{I}^*$ over time. In each training step k, we observe the average of the last 64 realized IC gains $\bar{\delta\mathfrak{I}}$. We apply the following gradient-based update rule to change the threshold θ:

$$\theta^k = \theta^{k-1} + (\bar{\delta\mathfrak{I}} - \delta\mathfrak{I}^*)\,.$$

Simply put, if the actual IC gain is too high, we increase θ to lower it in the future, and vice versa. To prevent pathological cases, we limit θ to the interval $[0.55, 1]$. It is initialized at the lower bound. The selection of the final pseudo-label matches section 5.3.3.2.

Since this method is fundamentally a solution to a control problem, it can suffer from similar failure cases. For example, the threshold could oscillate or react too slowly to changes in the hypothesis. To prevent such cases, an implementation could add a “learning rate” to the update rule (see section 2.1.3).

6. Experiments on Knowledge Integration and Imprecise Data

In this section, we present a selection of experiments concerning the semantic knowledge integration and imprecise data aspects of this thesis. Experiments regarding the lifelong learning aspects are detailed in the next chapter 7. We start by describing evaluation criteria, some of which can take concept hierarchies into account. Afterwards, we lay out technical details pertaining to the experimental setup and give an overview of generic and biology-specific benchmark datasets.

The first two experiments directly correspond to the proposed use cases for the hierarchical classifier (see section 5.2). We investigate integrating domain knowledge on benchmark data in section 6.3 to confirm the correct operation of our hierarchical classifier. Then, we evaluate learning from imprecise data in section 6.4, where we consider both CHILLAX (see section 5.3.2) and its self-supervised counterpart (see section 5.3.3).

A real-world application of imprecise data, where web crawling is used to improve moth species classification, is presented in section 6.5. We offer a detailed analysis of possible failure cases due to weak visual-semantic correspondence, faulty concept hierarchies and inappropriate imprecision modeling in section 6.6.

Finally, we summarize and discuss our findings in section 6.7.

6.1. Evaluation

In the following, we describe all *evaluation metrics* used in our experiments. Evaluation metrics are computed to empirically determine the quality of a given hypothesis on a validation set or held-out test set (see section 2.2.2.1). Similar to loss functions in section 2.1.1.7, we define an evaluation metric as a function $\mathcal{M} : \mathcal{H} \times \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}^+$. For a given validation set T with $|T| = m$, it is computed as follows, unless specified otherwise:

$$\mathcal{M}_T(h) = \frac{1}{m} \sum_{i=1}^{m} \left[\mathcal{M}(h, x_i, y_i) \right] .$$

Since our targeted machine learning task is almost always classification, we first list metrics relevant to classification in general. Second, we discuss metrics that are specific to classification with concept hierarchies. We also consider more general semantic measures, *i.e.*, semantic distances and similarities.

6.1.1. Classification

In section 2.1.1.1, we define classification as a prediction task, where the label set $\mathcal{Y}$ is a set $\mathcal{Y}^P$ of mutually exclusive classes, or concepts. We can further differentiate between *binary* classification, which considers only two classes, and *multi-class* classification.

6.1.1.1. Binary Classification

For problem with only two classes, *e.g.*, $\mathcal{Y}^P = \{-1, 1\}$, we commonly assign semantics to each class. We consider one *negative* and the other *positive*. A comparison between prediction and ground truth has four possible outcomes: true positive, true negative, false positive, and false negative. The second word in each phrase indicates the prediction, while true or false specify if it matches the ground truth. As evaluation metrics, we define the *true positive rate* TPR, the *false positive rate* FPR, and the respective TNR and FNR as follows:

$$\begin{aligned}
TPR_T(h) &= \frac{\sum_{i=1}^{m} \mathbb{1}[h(x_i) = 1 \wedge y_i = 1]}{\sum_{i=1}^{m} \mathbb{1}[h(x_i) = 1]}\,, \\
FPR_T(h) &= \frac{\sum_{i=1}^{m} \mathbb{1}[h(x_i) = 1 \wedge y_i = -1]}{\sum_{i=1}^{m} \mathbb{1}[h(x_i) = 1]}\,, \\
TNR_T(h) &= \frac{\sum_{i=1}^{m} \mathbb{1}[h(x_i) = -1 \wedge y_i = -1]}{\sum_{i=1}^{m} \mathbb{1}[h(x_i) = -1]}\,, \\
FNR_T(h) &= \frac{\sum_{i=1}^{m} \mathbb{1}[h(x_i) = -1 \wedge y_i = 1]}{\sum_{i=1}^{m} \mathbb{1}[h(x_i) = -1]}\,.
\end{aligned}$$

This nomenclature is used in machine learning, but also in other applications of statistics, *e.g.*, medicine. The true positive rate is also known as *sensitivity* and the true negative rate as *specificity* (*cf.* Yerushalmy 1947).

In retrieval contexts, but also for object detection, *precision* (PRE) and *recall* are common evaluation metrics (*cf.* Goodfellow, Bengio, and Courville 2016, p. 423):

$$\begin{aligned}
PRE_T(h) &= \frac{\sum_{i=1}^{m} \mathbb{1}[h(x_i) = 1 \wedge y_i = 1]}{\sum_{i=1}^{m} \mathbb{1}[h(x_i) = 1]}\,, \\
REC_T(h) &= \frac{\sum_{i=1}^{m} \mathbb{1}[h(x_i) = 1 \wedge y_i = 1]}{\sum_{i=1}^{m} \mathbb{1}[y_i = 1]} = TPR_T(h)\,.
\end{aligned}$$

Binary classifiers can predict probabilities instead of making simple decisions (see section 2.1.2.2). A threshold can then be applied to the predicted probabilities to reach a decision. This threshold can be optimized, *e.g.*, to prefer true positives or true negatives. Varying the threshold over its entire range and plotting the respective precision and recall for each threshold results in the receiver operater characteristics (ROC) curve (*cf.* ibid., p. 423). The area under this curve is denoted *average precision*.

6.1.1.2. Accuracy

For multi-class classification tasks, *i.e.*, where $|\mathcal{Y}^P| > 2$, the most common evaluation criterion is *accuracy* (ACC). It is defined on a per-sample basis as (*cf.* ibid., 103 sq.):

$$ACC(h, x, y) = \mathbb{1}[h(x) = y]\,. \tag{6.1}$$

The accuracy is the fraction of examples in the validation set that are predicted correctly. As such, it is exactly complementary to the definition of empirical risk in section 2.1.1.3.

If the classifier offers predicted probabilities for individual classes, the resulting ranking of classes can also be considered for evaluation. For example, the *top-k* accuracy indicates the fraction of examples where the ground truth class is in the top *k* of predicted classes (*cf.* Russakovsky et al. 2015).

Equation (6.1), which we also call *overall recognition rate*, does not consider the distribution of examples over classes in the validation set. However, if this distribution is very imbalanced or long-tailed, the *average recognition rate* (ARR) is a reasonable alternative:

$$ARR_T(h) = \frac{1}{|\mathcal{Y}^P|} \sum_{c \in \mathcal{Y}^P} \frac{\sum_{i=1}^{m} \mathbb{1}[h(x_i) = c \land y_i = c]}{\sum_{i=1}^{m} \mathbb{1}[y_i = c]} \, .$$

> Technically, loss functions could also be used to evaluate a hypothesis. Furthermore, comparing loss functions between training and validation sets (see section 2.2.2.1) can help detect instances of overfitting. However, the actual values of loss functions are rarely human-interpretable.

6.1.2. Hierarchical Classification

While hierarchical classifiers are often used to integrate domain knowledge in an effort to improve the non-hierarchical accuracy (see section 4.1), there are many evaluation criteria which can take a concept hierarchy into account. Some are applicable to non-hierarchical classifiers, and should always be considered, since qualitative differences between confused concepts are inherently semantic in nature (see section 4.1.5). In this section, we consider hierarchical equivalents to criteria from the previous section. Furthermore, we discuss several semantic measures, which are ways of comparing two concepts in the presence of a concept hierarchy.

6.1.2.1. Hierarchical Precision and Recall

The correct classification of an individual example can be framed as a retrieval problem. Here, the ground truth concept is represented by the set of all of its hypernyms (its intension, see section 3.1.2.2), similar to the internal labels in section 5.2.1. With a context $\mathcal{C}$, we define *hierarchical precision* (hPRE) and *hierarchical recall* (hREC) as (*cf.* Kiritchenko, Matwin, and Famili 2005):

$$\begin{aligned} hPRE(h, x, y) &= \frac{\{c \in \mathcal{C} : y \leq_{\text{is-a}} c\} \cap \{c \in \mathcal{C} : h(x) \leq_{\text{is-a}} c\}}{\{c \in \mathcal{C} : h(x) \leq_{\text{is-a}} c\}} \, , \\ hREC(h, x, y) &= \frac{\{c \in \mathcal{C} : y \leq_{\text{is-a}} c\} \cap \{c \in \mathcal{C} : h(x) \leq_{\text{is-a}} c\}}{\{c \in \mathcal{C} : y \leq_{\text{is-a}} c\}} \, , \end{aligned}$$

where $\leq_{\text{is-a}}$ is the reflexive hyponymy relation (see section 3.1). Note that there are competing definitions for hierarchical recall and precision, namely those proposed in Deng, Berg, and Fei-Fei (2011).

Because hierarchical precision and recall represent a trade-off similar to the choice of threshold in section 6.1.1.1, we further define a single metric combining the two values. The *hierarchical F1 score* (hF1) is defined as the harmonic mean of hPRE and hREC:

$$hF1(h,x,y) = \frac{2}{\frac{1}{hPRE(h,x,y)} + \frac{hREC(h,x,y)}{1}} .$$

6.1.2.2. Semantic Measures

Most of the aforementioned evaluation metrics boil down to a comparison between two elements of a label set, one provided by the hypothesis and one by the validation set. They are specifically designed to measure the performance of a model. However, there exist metrics that originally serve a different purpose, namely *semantic measures* (*cf.* Harispe et al. 2015). In section 5.1.1.2 we review IC, which is a semantic measure. For evaluation, we consider semantic measures that compare two concepts, which can be grouped into measures of *semantic distance* and *semantic similarity*.

The *Rada distance* is defined using a graph representation G of a concept hierarchy (*cf.* ibid., p. 61):

$$DIS_{Rada}(h,x,y) = d_G(h(x),y) . \tag{6.2}$$

It can be inverted into a measure of *semantic similarity* (*cf.* ibid., p. 61):

$$SIM_{Rada}(h,x,y) = \frac{1}{DIS_{Rada}(h,x,y)+1} . \tag{6.3}$$

Measuring the depth (see section 5.1.1.1) of the *lowest common subsumer* (LCS, the most specific hypernym shared by two concepts) offers an interesting interpretation. It is the depth, or specificity (see section 5.1.1.2) up to which the hypothesis is, on average, accurate. Alternatively, the *height* of the LCS, the shortest path distance to any leaf in G, can be considered. In Verma et al. (2012b), it is divided by the largest possible height to normalize the metric, which indicates dissimilarity. We list further semantic measures in section 6.6.1.1.

6.2. Experimental Setup

Several experiments in the following sections share common elements concerning their setup. In this section, we describe the datasets, neural network architectures and pre-processing procedures used in our experiments.

6.2.1. Datasets

As described in sections 2.3.3.1 and 2.3.3.2, acquiring sufficient amounts of training data for experiments with large neural networks is often cost-prohibitive. Hence, we rely on publicly available benchmark datasets. These datasets include evaluation protocols for easier comparison between a large range of methods. Furthermore, the label acquisition process is often described transparently and in a replicable manner. In the following, we review several benchmark datasets, which mostly represent classification tasks. We also describe datasets with biological applications, *i.e.*, species classification.

6.2.1.1. General Benchmark Datasets

CIFAR-10 and CIFAR-100 Based on the "80 Million Tiny Images" dataset presented in Torralba, Fergus, and Freeman (2008), the CIFAR-10 and CIFAR-100 benchmarks (presented in Krizhevsky and Hinton 2009) are of much smaller scale. Each dataset consists of 60 000 images 32×32 pixels in size. CIFAR-10 and CIFAR-100 differentiate between 10 and 100 concepts, respectively.

ImageNet The name "ImageNet" describes different databases depending on the context. In general, it is simply a very large database of medium- to high-resolution images associated with WordNet synsets (see section 3.3.1.1). Its initial release in 2009, presented in Deng et al. (2009), consists of approximately 3 200 000 images annotated with 5247 different synsets.

ImageNet is accompanied by a yearly challenge (the ImageNet Large-Scale Visual Recognition Challenge, ILSVRC) with a variety of benchmarks including classification and object detection tasks. A review of several years of challenges is presented in Russakovsky et al. (2015). The 2012 iteration is particularly popular, as one of the entries, Krizhevsky, Sutskever, and Hinton (2012), is considered the "breakthrough" of modern CNN approaches. Hence, the name ImageNet often refers to the ILSVRC 2012 classification benchmark dataset, which is also named ImageNet-1k. This 1000-class benchmark dataset ships with 1 200 000 training images and 50 000 validation images. The labels of the 100 000 held-out test images remain private.

PASCAL VOC While classification is an attractive task due to its relative simplicity both in implementation and evaluation, there are object detection benchmarks as well. "PASCAL Visual Object Classes" (VOC, Everingham et al. (2010, 2015)) is a yearly challenge and benchmark dataset for a variety of computer vision tasks, including object detection. The object detection benchmark dataset of the 2012 release is a common choice for evaluation. It considers only 20 object classes, but contains 11 540 medium-resolution images which are annotated with 27 450 object instances. Each instance is accompanied by a "difficult" flag, which is set if the object is partially occluded or otherwise difficult to detect.

6.2.1.2. Biological Benchmark Datasets

We consider biological datasets specifically for two main reasons. First, there is an agreed upon concept hierarchy, at least for species classification. Hence, even if a dataset does not include a concept hierarchy, it can be acquired from a biological knowledge base (see section 3.3.3.1). Second, biodiversity monitoring is a globally important application which can benefit from semantic knowledge integration. Furthermore, it has several traits that make it an excellent fit for lifelong learning, which we explore in chapter 7.

CUB-200 Presented in Welinder et al. (2010), the "Caltech-USCD Birds 200" (CUB-200) dataset is a common benchmark for methods of fine-grained recognition (see section 4.2.2). The goal is to classify 200 species of birds found mostly in North

America. There are approximately 6000 medium-resolution images in total. In addition to labels, the authors provide bounding box annotations as well as attributes. They warn that an overlap may exist between the test set of CUB-200 and the training set of ImageNet-1k (see section 6.2.1.1).

NABirds A more large-scale bird classification challenge is posed in Van Horn et al. (2015) under the name "NABirds". It is more fine-grained as it requires discriminating between juvenile and adult birds, not only species. The 555 classes contain 400 species of birds, which are represented by 48 562 images. While Amazon Mechanical Turk is a common way of acquiring annotations, NABirds is labeled (for free) by citizen scientists. The authors claim reduced label noise compared to CUB-200, which they measured by collaborating with domain experts.

iNaturalist Similar to ImageNet, the name "iNaturalist" has different meanings depending on the context. In general, iNaturalist[1] is a citizen science project and social network to share observations of animals and discuss them with fellow hobbyists and experts. The project is aimed towards supporting researchers in biodiversity monitoring and conservation efforts. iNaturalist also refers to yearly releases of large-scale fine-grained visual recognition (see section 4.2.2) challenge datasets as described in Van Horn et al. (2018). For example, the 2017 release consists of approximately 675 000 images which should be classified into 5089 different species. This dataset also exhibits considerable class imbalance, which is attributed to the observation frequency of the respective species.

6.2.2. Deep Neural Networks

This section offers technical details concerning the deep neural networks used in our experiments. We describe neural network architectures as well as initialization and optimization choices.

6.2.2.1. Architectures

ResNet Residual networks (*cf.* He et al. 2016a,b), which we describe in more detail in section 2.4.3.3, are a common choice for contemporary deep learning applications. There are several available implementations, *e.g.*, Keras[2], TensorFlow[3], PyTorch[4], as well as the original Torch code[5]. Because of its relevance for potential applications and widespread use, we select the ResNet-50 architecture described in He et al. (2016b) for our experiments, unless specified otherwise. This architecture is sometimes denoted "ResNet-50 V2" to differentiate it from "V1", an earlier design from He et al. (2016a). When configured for ImageNet-1k, it has 25 613 800 individual parameters.

---

[1] `https://www.inaturalist.org/`

[2] `https://keras.io/`

[3] `https://www.tensorflow.org/`

[4] `https://pytorch.org/`

[5] `https://github.com/facebookarchive/fb.resnet.torch`

6.2.2.2. Pre-training and Fine-tuning

While large benchmark datasets such as ImageNet-1k (see section 6.2.1.1) are sufficient to train architectures such as the aforementioned ResNet-50 without significant overfitting, smaller datasets, *e.g.*, CUB-200 (see section 6.2.1.2), are not. This statement assumes that training is performed "from scratch", *i.e.*, with parameters initialized randomly (see section 2.4.3.1). However, it is common practice to initialize parameters by copying an already converged hypothesis (*cf.* Goodfellow, Bengio, and Courville 2016, 323 sqq.). This process is termed *fine-tuning* a *pre-trained* model. Using this method, the sample complexity of the hypothesis is reduced because the initialization provides an inductive bias, which is assumed to be helpful. Smaller amounts of training data are necessary to reach a certain true risk if the dataset used to optimize the pre-trained model contains relevant knowledge. Pre-training can also be useful if performed multiple times (*cf.* Cui et al. 2018).

6.2.2.3. Learning Rate Schedules

SGD (see section 2.1.3) does not converge under the same circumstances as regular gradient descent. The minibatch selection introduces noise such that even if the loss function w.r.t. the whole training data is minimal, the gradient w.r.t. to the minibatch is not zero. However, convergence can be improved using a *learning rate schedule*, such that each step k uses a different learning rate $\eta^{(k)}$. If, for $k \rightarrow \infty$, the sum over all $\eta^{(k)}$ diverges, but the sum over all $(\eta^{(k)})^2$ does not, convergence of SGD is guaranteed (*cf.* Goodfellow, Bengio, and Courville 2016, 294 sq.). For example, the learning rate can be reduced linearly until a minimum is reached (*cf.* ibid., 295, eq. 8.14).

Stochastic gradient descent with warm restarts (SGDR, Loshchilov and Hutter (2017)) is specifically tailored to CNNs. The learning rate is decayed following a cosine function. However, when the zero crossing of the cosine is reached, the learning rate is reset to its initial value and the decay begins again. This is denoted a *warm restart* because the learning rate is "restarted", but the parameters $\boldsymbol{\theta}$ are kept.

6.2.3. Data Processing

The following paragraphs explain the transformations which an image undergoes from its original state as part of the training data towards a compatible input to a neural network.

6.2.3.1. Augmentation

In section 2.3.4.2 we discuss several invariances and covariances of computer vision tasks. *Data augmentation* (*cf.* Goodfellow, Bengio, and Courville 2016, 240 sqq.) uses the knowledge of these invariances to "augment" the training data with transformed images. For example, if a given classification task is known to be invariant to a horizontal mirror transform of the coordinates, the training data can be doubled by adding a mirrored version of each example. The exact invariances are always task-specific and are a form of domain knowledge. A drawback of data augmentation is that each invariance has to be learned from the augmented training data, which requires time. However, the alternative of constructing a hypothesis class that has the correct invariance, is not always feasible. Our framework allows for the following transforms:

- Horizontal and vertical mirroring with a 50 % probability,
- Rotation around the center by a random angle $\sim \mathcal{U}[0, 2\pi]$,
- Cropping of a region randomly sized and positioned, which may also extend outside the original image,
- Gray value transforms to randomly increase or decrease brightness and contrast,
- Color transforms to randomly variate hue and saturation.

Unless specified otherwise, we only perform horizontal mirroring, *i.e.*, exchange left and right.

6.2.3.2. Image Geometry and Representation

While digital images are discrete in both range and domain (see section 2.3.1.1), the neural networks we apply in our experiments expect a continuous range (see section 2.4). Continuity is also a requirement of gradient-based optimization (see section 2.1.3). In this section, we briefly discuss the pre-processing necessary in order to meet these requirements.

First, we map the original discrete $\mathbb{F}^{256\times 3}$ color space to a floating-point representation of $[0,1]^3$ by type casting and dividing by 256. We then subtract the mean color over the entire training data and divide by the standard deviation to "whiten" the range of the images. The image is then scaled to a size slightly larger than the neural network's input. For example, if the neural network expects an input size of 224×224 pixels, we scale each image to 256×256 using bilinear interpolation. The final size is obtained by cropping a fitting area at a random location such that the cropped region is completely inside the image (*cf.* Krizhevsky, Sutskever, and Hinton 2012). We only apply random cropping during training. For predictions, we crop the exact center of the image.

6.3. Knowledge Integration on Benchmark Datasets

Partial results of the work presented in this section are published in Brust, Barz, and Denzler (2021b) and Brust and Denzler (2019a).

This experiment should serve as an entry point to the experimental section as a whole. It concerns the hierarchical classifier proposed in section 5.2, specifically the deep learning implementation detailed in section 5.2.4. We observe the change in accuracy (w.r.t. a validation set) of using the hierarchical classifier instead of the conventional one-hot softmax (OHSM) formulation (see sections 2.4.1.3 and 2.4.1.4).

For the purpose of this thesis, we integrate domain knowledge into a system to enable the capability of learning from imprecise data, which is not possible with an

OHSM classifier. Nevertheless, it is important to study the effect of the method itself to validate its utility and also to more accurately gauge the "value" of imprecise data on its own. More specifically, the goal is not to improve the performance of a model by using our hierarchical classifier to integrate domain knowledge. It should be noted that there are several methods that can improve performance (see section 4.1). However, they support neither imprecise data nor concept hierarchies that are DAGs without non-trivial additional assumptions (see section 6.4.2.4).

Hypotheses Our setup combines several individual hypotheses, which are isolated in subsequent experiments. Interestingly, there are hypotheses to explain a negative, a positive and also no effect of using the hierarchical classifier. If the assumptions in section 5.2.2, which are derived from the concept hierarchy, hold true for the specific training data, accuracy should improve. Because of the structure of the hierarchical classifier, the model can no longer make predictions that violate the assumptions, while the OHSM classifier can. It also does not have to learn the relations between concepts from the training data, which might not be sufficiently representative.

There is also a reason to expect no effect at all. In Bilal et al. (2018), it is shown that CNNs can already learn the concept hierarchy from training data without explicit integration (see section 4.1.5).

Furthermore, the hierarchical classifier adds error sources that are not present in OHSM. The assumptions in section 5.2.2 could be faulty, *e.g.*, because of too weak a correspondence between visual and semantic features (see section 6.6.1). Moreover, the concept hierarchy itself could contain erroneous entries (see section 6.6.2). Finally, the hierarchical classifier does not assume mutual exclusivity between sibling concepts as that is not trivial with DAG hierarchies, although the assumption would hold for almost all concepts. All the aforementioned causes could result in decreased accuracy.

6.3.1. Setup

For clarity, we divide the experiment into two scales of concern w.r.t. computational resources (see also section 2.3.3.2). Small-scale experiments can perform an individual run in less than a day, where a "run" consists of a single full training (until convergence) and multiple validations of a single neural network. The small scale is also characterized by constrained computing resources in terms of working memory (4 GiB of VRAM), leading to small image dimensions and minibatch sizes (see section 2.1.3.1). An individual run of our large-scale experiments completes in less than a week, and has access to reasonable compute and memory amounts, *i.e.*, a single GPU and 16 GiB of VRAM. It should be noted that our large scale is still several orders of magnitude from the scale of commercial research organizations such as Facebook AI[6] and OpenAI[7].

We perform the small-scale experiments because an interesting side effect of domain knowledge integration could be that it counteracts losses incurred from the

---

[6] `https://ai.facebook.com/`

[7] `https://openai.com/`

Table 6.1.: Small-scale benchmark comparison of hierarchical classifier and baseline. For each dataset, we report the accuracy in percent on the respective validation set. Table compiled of data from Brust and Denzler (2019a).

			Accuracy (%)	
Dataset	$\lvert\mathcal{Y}^P\rvert$	$\lvert\mathcal{C}\rvert$	Baseline	HC
ImageNet-1k	1000	1860	49.00 ± 0.33	**54.20** ± 0.04
NABirds (center crops)	555	1010	56.50 ± 0.49	**61.90** ± 0.27
CIFAR-100	100	267	**55.40** ± 0.84	54.60 ± 1.03

Table 6.2.: Large-scale benchmark comparison of hierarchical classifier and baseline. For each dataset, we report the accuracy and hierarchical F1 score (hF1) in percent on the respective validation set. Table compiled of data from Brust, Barz, and Denzler (2021b).

	Accuracy (%)		hF1 (%)	
Dataset	Baseline	HC	Baseline	HC
ImageNet-1k	**65.00**	62.52	**87.70**	86.66
NABirds	**82.72** ± 0.14	81.40 ± 0.17	**91.15** ± 0.07	90.67 ± 0.08

constrained scale. Furthermore, the small scale should be more representative of actual field applications in biodiversity research, *i.e.*, on edge devices (see section 2.3.3.2). The large-scale experiments are run to determine if it is possible to compete at already very high accuracy levels. However, these experiments are still constrained by available resources and time much more so than the state-of-the-art models such as EfficientNet (Tan and Le 2019) or transformers (Dosovitskiy et al. 2021).

Our baseline is an unmodified, conventional OHSM classifier based on a ResNet architecture (see section 6.2.2.1), depending on the dataset. The hierarchical classifier setup is identical, except for the last layer of the neural network, which is replaced as described in section 5.2.4, together with the loss function and label encoding. As benchmark datasets, we use CIFAR-100 (small-scale only), NABirds, and ImageNet-1k, which are reviewed in detail in section 6.2.1.1. All hyperparameters and further settings are provided in the appendix, in table A.1 for the small-scale experiments and table A.2 for large-scale. The tables also list the number of experimental runs used to determine the mean and standard deviation of our measurements. This number is determined by the respective scale and expected runtime.

The "iNaturalist 2017" initialization specified in table A.2 refers to the results from Cui et al. (2018), which are available online[8].

6.3.2. Results

In the following, we review the results of this experiment. We perform a quantitative evaluation using the measures discussed in section 6.1. For the small-scale experiments, we focus on accuracy and efficiency. For the large-scale counterparts,

---

[8] `https://github.com/richardaecn/cvpr18-inaturalist-transfer/`

we provide a more detailed analysis of the models' predictions, including hierarchical measures (see section 6.1.2). We also perform a qualitative analysis of the mispredictions.

The results and their consequences as well as possible explanations are discussed in section 6.7.1.1.

6.3.2.1. Small-Scale Results

Table 6.1 shows the results of our small-scale experiment. When the scale is restricted, the hierarchical classifier outperforms the OHSM classifier baseline on both NABirds and ImageNet-1k by a margin of approximately five percent points. However, on CIFAR-100, the baseline achieves a marginally higher accuracy by less than one percent point. We further observe that the relative improvement in accuracy scales according to the number of concepts in the respective dataset.

Since the small-scale experiment should represent limited resources, the efficiency of training is an important aspect. Specifically, the fewer optimization iterations required to reach a certain reasonable "working" performance level, the better. In a lifelong learning system (see section 7.1), where incremental learning is run repeatedly, faster convergence is crucial. Thus, such a system benefits strongly from an efficient classifier.

We first consider CIFAR-100, where both the baseline and our hierarchical classifier reach comparable final results, but at notably different speeds. After the first 500 iterations, the accuracy of the hierarchical classifier is already 10.70 %, while the OHSM model reaches only 2.80 %. It takes 2100 iterations in total to manage a comparable result. Hence, the hierarchical classifier might be a better choice in scenarios where training time is limited, even though the baseline reaches a higher final accuracy.

Next, we observe the learning pace on NABirds, which has around five times the number of concepts compared to CIFAR-100. The hierarchical classifier predicts the validation set with 10.60 % accuracy after the first 5000 iterations. At the same time, the OHSM baseline's accuracy is only 0.40 %. The baseline requires 21 000 steps in total to match the hierarchical classifier's accuracy at 5000 steps. The resulting "speed-up" is the same as for CIFAR-100.

The final observations concern ImageNet-1k, which again doubles the scale of NABirds. We take a first validation measurement after 31 250 steps, where the hierarchical classifier shows an accuracy of 28.90 %, while the baseline reaches 20.50 %. In this setting, the baseline reaches a comparable measure after 62 500 steps. As with NABirds, it never "overtakes" the hierarchical classifier. However, the implied speedup is smaller than on the other datasets. A discussion of these findings is offered in section 6.7.1.1.

Concise Results

Small scale: The hierarchical classifier requires fewer training steps to reach a certain accuracy than the OHSM baseline. It also outperforms the baseline on two out of three datasets.

6.3.2.2. Large-Scale Results

The large-scale experimental results are presented in table 6.2.

We first compare accuracies on the ImageNet-1k benchmark dataset. The OHSM baseline outperforms the hierarchical classifier with a final accuracy of 65.00 % vs 62.52 %. For ImageNet specifically, we can also consider the top-5 accuracy (see section 6.1.1), which is used for the accompanying challenge. In terms of top-5 accuracy, the baseline achieves 86.68 %, while the hierarchical classifier reaches 83.55 %. Furthermore, the hierarchical F1-Score (see section 6.1.2) is marginally in favor of the baseline. It produces an hF1 of 87.70 %, compared to the hierarchical classifier's 86.66 %. Overall, there are only small differences in performance between the OHSM model and the hierarchical classifier. However, these differences are in favor of the baseline, and in contrast to the small-scale results, which overall show the hierarchical classifier as superior.

We also look at the NABirds dataset as a more fine-grained scenario, which is also more representative of the envisioned biodiversity research applications. Again, the OHSM baseline slightly outperforms the hierarchical classifier. In terms of accuracy, the results are 82.72 % against 81.40 % on the validation set. The hierarchical F1 values are even closer at 91.15 % to 90.67 %. Overall, both models exhibit comparable performance, which is again marginally in favor of the OHSM baseline. We discuss the results in section 6.7.1.1.

Concise Results

Large scale: The OHSM baseline outperforms the hierarchical classifier in all settings, but only marginally. This holds true for both generic and hierarchical evaluation measures.

6.3.2.3. Qualitative Analysis of Large-Scale Results

Even though the OHSM baseline model is not equipped with a concept hierarchy, we can nevertheless apply hierarchical evaluation criteria to its predictions (see section 6.1.2). For example, there is research indicating that hierarchical classifiers make "better mistakes" (*cf*. Bertinetto et al. 2020). Such statements could not be made if it were impossible to measure the quality of mistakes made by the non-hierarchical model. In the following, we analyze predictions and mispredictions according to hierarchical criteria to determine whether this research also applies to our hierarchical classifier. However, it is also claimed that CNNs learn concept hierarchies by themselves (*cf*. Bilal et al. 2018), which could be interpreted as a contradictory statement. Related analysis works are discussed in section 4.1.5.

In the previous section, we already discuss the hierarchical F1-scores. We now compare the lowest common subsumer (LCS) depth as explained in section 6.1.2.2. This measure is slightly more interpretable for applications, *e.g.*, as in "this model is, on average, accurate up to the species level".

On ImageNet-1k, the mean LCS depth between prediction and ground truth on the complete validation set is 9.92 for the baseline, and 9.80 for the hierarchical classifiers. These measurements are very close. With NABirds, the difference is even smaller, both absolutely and relatively, at 4.12 for the baseline against 4.10. Note that the scale of the numbers themselves depends explicitly on the concept hierarchy. As such, it cannot be compared across datasets, or even across different concept hierarchies describing the same dataset.

To gain further insight, we consider the same LCS depth metric, but as an average of only the mispredictions, instead of the whole validation set. Even if one method produces more mispredictions, they could still be of a different quality, *i.e.*, "better". Starting with ImageNet-1k, LCS depth between mispredictions and the respective ground truth is 7.22 for the baseline. For the hierarchical classifier, it is 7.16, which is still slightly worse. On NABirds, we observe the opposite situation. The misprediction LCS depth achieved by the baseline is 2.19. The hierarchical classifier makes marginally better mistakes at 2.24, which we discuss in section 6.7.1.1.

Concise Results

Large scale: The hierarchical classifier makes "better mistakes" on NABirds, but it does not on ImageNet-1k.

6.4. Imprecise Data from Benchmark Datasets

Partial results of the work presented in this section are published in Brust, Barz, and Denzler (2021b, 2022).

This experiment is the first in this thesis concerning imprecise data. Imprecise data is the main motivation for developing the probabilistic hierarchical classifier outlined in section 5.2. We now consider CHILLAX, described in section 5.3.2, built on top of this classifier to accept imprecise data. Furthermore, we test the self-supervised variant proposed in section 5.3.3, which addresses a number of drawbacks of CHILLAX.

As in the previous experiment, we rely on benchmark datasets (see section 6.2.1.1). We select NABirds for its relevance to our targeted biodiversity applications, and ImageNet-1k such that we can compare CHILLAX to HEX (*cf.* Deng et al. 2014), another method that is compatible with imprecise data.

Both datasets initially contain high-quality annotations. We obtain imprecise data by purposely reducing the precision of the labels in the training data. Moreover, we introduce further label noise by performing random confusions.

Our expectation is that the imprecise data can be leveraged by CHILLAX, resulting in higher performance compared to classifiers that have no access to this data simply as a consequence of the higher number of samples. However, compared to precise examples, we expect the added value of imprecise data to be slightly lower. Hence, we expect the reduction of precision to lead to lower accuracies for all methods, but to affect CHILLAX the least.

We detail the noise process in the following.

6.4.1. Simulating Label Noise

When viewing an annotator-provided label as a measurement, there are two main aspects of concern, namely accuracy and precision. We offer a detailed description of these measures in section 5.1. In our experiments, our goal is to study the effects of each aspect on the resulting accuracy of CHILLAX independently.

To this end, we consider synthetic labels, for which known-good annotations from benchmark datasets are an ideal starting point. We can control the amount of accuracy and precision directly by adapting our modifications. Only the training data is modified. We rely on the unaltered validation set to provide a fair comparison between different qualities of annotations and methods.

6.4.1.1. Introducing Imprecision

First, we describe how we artificially reduce the precision of annotations. For simplicity, we define precision in terms of the depth of a concept in the concept hierarchy. This depth-based model is described in more detail in section 5.1.1.1. We consider the following *noise models* from the aforementioned section:

- No imprecision,
- *Poisson distribution*: labels by volunteers, parameterized by λ,
- *Geometric distribution*: labels from web crawling, parameterized by q,
- *Deng et al. (2014)*: reduce depth by one, parameterized by p.

The depth reduction of the training data is implemented using algorithm 1. We validate the aforementioned noise models on real-world data in section 6.6.3.

6.4.1.2. Adding Inaccuracy

Second, we assume that imprecise data is an expression of annotator uncertainty, *e.g.*, because of a lack of expertise. Hence, annotators only label as precisely as they are comfortable. However, they could still make mistakes in which they are very confident, while most other humans might not agree on the decision. By introducing random confusions independent of the structure of the concept hierarchy, we can analyze the robustness of CHILLAX to this variant of label noise.

In our implementation, we apply random confusions before we perform the depth reduction described in the previous section. Since our original training data is exclusively precise, we only consider random confusions between precise concepts, *i.e.*, leaf nodes, with equal probabilities. Note that it is possible for the subsequent depth reduction to "correct" a number of the modified labels. This situation occurs when the original label and the modified version have a common subsumer (see section 6.1.2), and the subsequent depth reduction results in this subsumer or an even less precise concept.

Table 6.3.: Remaining fraction of precise examples (%) and average IC in the modified NABirds training set after applying the specified noise models.

Deng et al. p	0.99	0.95	0.9	0.5
Precise Examples	1.00	5.00	10.00	50.00
IC	0.83	0.84	0.85	0.90
Geometric q	0.5	0.8	0.9	0.95
Precise Examples	9.60	45.90	69.00	83.40
IC	0.29	0.63	0.79	0.88
Poisson λ	1.0	2.0	3.0	4.0
Precise Examples	4.80	22.70	45.90	65.90
IC	0.32	0.58	0.75	0.85

Table 6.4.: Remaining fraction of precise examples (%) and average IC in the modified ImageNet-1k training set after applying the specified noise model.

Deng2014 (p)	0.99	0.95	0.9	0.5
Precise Examples	1.00	5.00	10.00	50.00
IC	0.79	0.79	0.80	0.86

6.4.1.3. Statistical Properties of Simulated Labels

Before conducting the actual experiment, we analyze the training data obtained from the aforementioned modifications. This gives further context to interpret the results in the following experiments. We consider two statistical aspects. First, the number or fraction of precise labels that remain in the training data after modification. Initially, it is 100 %. Second, the average intrinsic IC of the labels as described in section 5.1.1.2. On NABirds, it is 0.97 initially, while on ImageNet-1k, it is 0.93. Both values are lower than 1.00 because the concept hierarchies are not balanced. Tables 6.3 and 6.4 show the statistics on the NABirds and ImageNet-1k datasets, respectively.

6.4.2. Supervised CHILLAX

We first consider the purely and fully supervised variant of CHILLAX, which is developed in section 5.3.2 on top of the hierarchical classifier from section 5.2. There are three aspects of interest in this investigation. First, the effect of increasingly imprecise data on the accuracy of CHILLAX, and two baselines, w.r.t. the unmodified validation set. Second, the relationship between the average intrinsic IC of the training data and the resulting accuracy, which could help identify a trade-off between precision and accuracy. Finally, the comparison to HEX proposed in Deng et al. (ibid.), which is, to the best of our knowledge, the only competitor to CHILLAX.

Table 6.5.: Large-scale benchmark comparison of CHILLAX and baselines on NABirds. We report the accuracy in percent on the validation set. There is no imprecision, inaccuracy as indicated. Table compiled of data from Brust, Barz, and Denzler (2021b).

Inaccuracy	—	1 %	10 %
Leaves only	**82.80** ± 0.18	81.96 ± 0.19	77.19 ± 0.19
Random leaf	82.64 ± 0.11	**82.09** ± 0.10	**77.35** ± 0.15
CHILLAX	81.40 ± 0.17	80.91 ± 0.19	75.28 ± 0.39

6.4.2.1. Setup

Our investigation focuses on the NABirds and ImageNet-1k benchmark datasets (see section 6.2.1.1). The respective hyperparameters are detailed in the appendix, in table A.3.

Because results for HEX are only available on ImageNet-1k, we provide two further baselines based on heuristics that make imprecise data precise again. They are chosen to represent annotators which are only able to make an imprecise decision, but are forced to give a precise annotation. An OHSM classifier then learns the resulting training data. The *leaves only* baseline ignores all elements of the training data that are not precise, *i.e.*, whose labels are not in $\mathcal{Y}^P$. Depending on the noise model, this might only leave a small amount of examples (see section 6.4.1.3 for exact numbers). Alternatively, the *random leaf* baseline selects a random element of $\mathcal{Y}^P$ for each imprecise label, such that the leaf node is subsumed by the label. For very imprecise labels, this can introduce a significant amount of noise. However, for slight imprecision such as occurs in the experimental protocol of Deng et al., it can be a reasonable choice as the probability of randomly choosing the correct label is up to 50 %.

Note that both baselines only transform imprecise data. Hence, if all labels are precise, they are identical and equal to an OHSM classifier.

6.4.2.2. Large-Scale Results

For our first large-scale evaluation, we focus on the results on the NABirds dataset (see section 6.2.1.2). The results on the ImageNet-1k dataset are presented in section 6.4.2.4. As a point of reference, table 6.5 shows the accuracies produced by CHILLAX and the two baselines when there is no imprecision and no inaccuracy present. Both baselines have an advantage of approximately one percent point in accuracy over CHILLAX.

This is a reasonably small difference as long as it can be overcome in the presence of imprecise data. We introduce imprecision as described in section 6.4.1.1, first without additional inaccuracy, and obtain the results shown in table 6.6. Starting with the imprecision model proposed by Deng et al., we observe that CHILLAX outperforms both baselines by a margin of several percent points. Moreover, there is a substantial difference in performance between the baselines, which results from the imprecision model. Since the depth of labels is reduced by one at most, selecting a random leaf is not unreasonable. However, attempting to learn from only precise examples when

Table 6.6.: Large-scale benchmark comparison of CHILLAX and baselines on NABirds. We report the accuracy in percent on the validation set. There is imprecision as indicated, no inaccuracy. Table compiled of data from Brust, Barz, and Denzler (2021b).

Deng et al. p	0.99	0.95	0.9	0.5
Leaves only	9.00 ± 0.58	28.21 ± 0.64	45.57 ± 1.61	76.26 ± 0.34
Random leaf	60.91 ± 0.65	61.47 ± 0.45	63.48 ± 0.19	74.34 ± 0.47
CHILLAX	**63.24** ± 0.44	**70.37** ± 0.65	**75.10** ± 0.43	**80.94** ± 0.10
Geometric q	0.5	0.8	0.9	0.95
Leaves only	42.89 ± 0.58	75.16 ± 0.29	**79.58** ± 0.09	**81.38** ± 0.23
Random leaf	12.49 ± 0.39	51.37 ± 0.48	67.92 ± 0.24	75.48 ± 0.27
CHILLAX	**48.86** ± 0.97	**75.57** ± 0.10	79.07 ± 0.28	80.28 ± 0.11
Poisson λ	1.0	2.0	3.0	4.0
Leaves only	26.47 ± 0.82	61.87 ± 0.55	74.87 ± 0.32	79.07 ± 0.23
Random leaf	11.15 ± 0.37	36.82 ± 0.41	58.98 ± 0.51	70.57 ± 0.27
CHILLAX	**42.87** ± 0.38	**70.13** ± 0.24	**77.70** ± 0.31	**80.11** ± 0.15

they represent 1 % of the training data predictably results in a comparatively small accuracy.

The geometric noise model, which is designed to represent the distribution of precision obtained by crawling the web for annotated training data, leads to slightly different results. For $q = 0.9$ and $q = 0.95$, the *leaves only* baseline outperforms the competition including CHILLAX. In these cases, the training data consists of 69.00 % and 83.40 % precise examples (see table 6.3). Where imprecision is stronger, *i.e.*, $q = 0.5$ and $q = 0.8$, CHILLAX again performs best, although only marginally at $q = 0.8$. The *random leaf* baseline performs consistently worst.

Moving on to Poisson-distributed imprecision modeling the expertise and behavior of volunteer annotators, we observe similar characteristics. In this situation, CHILLAX reaches the highest accuracies in all settings, but again has only a marginal advantage when $\lambda = 4$. *Leaves only* is only competitive at this highest level of precision, while *random leaf* is consistently the worst performer again.

Concise Results

No inaccuracy: Overall, CHILLAX compares favorably to the two baselines *leaves only* and *random leaf*. However, the specific advantage depends on the amount of imprecision and fades in cases where the training data is largely precise.

We now introduce inaccuracy (see section 6.4.1.2), starting with random confusions of 1 % of the training data. Table 6.5 shows all accuracies obtained from fully precise, but inaccurate training data. Compared to the unmodified training data, the

Table 6.7.: Large-scale benchmark comparison of CHILLAX and baselines on NABirds. We report the accuracy in percent on the validation set. Imprecision as indicated, 1 % inaccuracy.

Deng et al. p	0.99	0.95	0.9	0.5
Leaves only	8.73 ± 0.66	27.09 ± 1.53	45.92 ± 1.47	75.64 ± 0.23
Random leaf	60.54 ± 0.55	61.77 ± 0.24	63.04 ± 0.37	73.81 ± 0.19
CHILLAX	**62.99** ± 0.32	**69.53** ± 1.17	**73.87** ± 0.59	**80.25** ± 0.15
Geometric q	0.5	0.8	0.9	0.95
Leaves only	40.47 ± 0.62	74.70 ± 0.16	**78.90** ± 0.26	**80.89** ± 0.21
Random leaf	11.96 ± 0.40	51.11 ± 0.09	67.96 ± 0.51	74.92 ± 0.31
CHILLAX	**47.27** ± 0.72	**74.72** ± 0.21	78.17 ± 0.10	79.75 ± 0.10
Poisson λ	1.0	2.0	3.0	4.0
Leaves only	25.29 ± 1.64	60.64 ± 0.58	74.22 ± 0.12	78.51 ± 0.33
Random leaf	10.89 ± 0.12	36.30 ± 0.77	58.38 ± 0.29	70.34 ± 0.21
CHILLAX	**41.96** ± 0.29	**69.37** ± 0.40	**76.79** ± 0.16	**79.09** ± 0.17

accuracies are reduced by approximately one half percent point, while the distance between CHILLAX and the baselines remains small. Combining 1 % inaccuracy and our three imprecision models, we obtain the results presented in table 6.7.

Concerning the noise model by Deng et al., the results remain qualitatively unchanged. In the geometric case, we observe similar results, except for $q = 0.8$, where the marginal advantage of CHILLAX without inaccuracy is reduced to almost zero. Poisson-distributed imprecision with inaccuracy again exhibits no qualitative difference compared to fully accurate training data.

Concise Results

Slight inaccuracy (1 %): CHILLAX again compares favorably to the two baselines *leaves only* and *random leaf*. The specific advantage still depends on the amount of imprecision, but is marginally smaller compared to full accuracy.

We finally investigate a scenario with 10 % random confusions, which is representative of extremely noisy label sources such as web crawling or citizen science projects with low participation. The results are presented in tables 6.5 and A.4. Qualitatively, they represent a continuation of the trend we observe when moving from 0 % to 1 % inaccuracy. Specifically, the geometric nose model at $q = 0.8$ exhibits a new order, where the *leaves only* baseline outperforms both CHILLAX and *random leaf*. In addition, we observe a similar result for *leaves only* in Poisson distributed imprecision. For $\lambda = 4$, *leaves only* obtains the highest accuracy of 73.06 % compared to CHILLAX at 72.81 %. We discuss our findings in section 6.7.1.2.

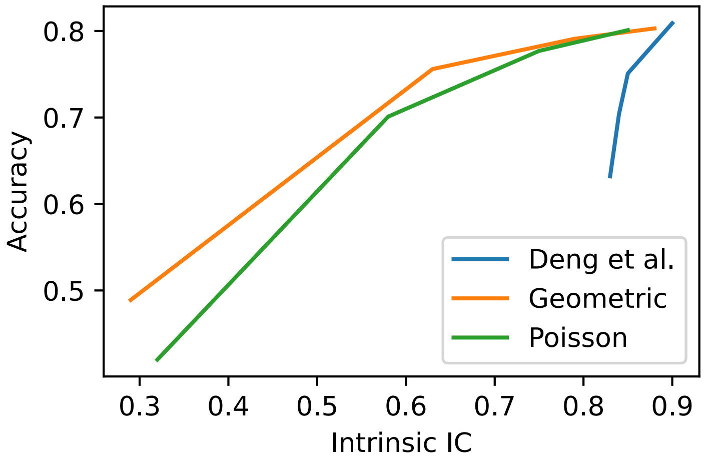


Figure 6.1.: Intrinsic IC in relation to accuracy (%) of CHILLAX on NABirds validation set obtained by applying different models and amounts of imprecision on training set.

Concise Results

Strong inaccuracy (10 %): CHILLAX still compares favorably to the two baselines *leaves only* and *random leaf*. The overall advantage is reduced further compared to 1 % inaccuracy, but is still substantial for Poisson-distributed imprecision as well as according to Deng et al. When imprecision is distributed geometrically, CHILLAX only performs best for $q = 0.95$.

6.4.2.3. Intrinsic Information Content vs. Accuracy

Figure 6.1 considers the average intrinsic IC of the training data after imprecision is applied. Empirical values and the measurement process are detailed in section 6.4.1.3. The individual intrinsic IC values are compared to the validation accuracies obtained by CHILLAX after learning from the respective modified training data.

We observe a similar correlation between the fundamentally different geometric and Poisson noise models. The model of Deng et al., however, produces substantially different curves in that a given value of intrinsic IC corresponds to lower accuracies than for the other two noise distributions. We discuss the consequences in section 6.7.1.2.

Concise Results

Accuracy and intrinsic IC: The efficiency implied through certain accuracies CHILLAX reaches by learning from training data with a specific IC is similar for imprecision with Poisson and geometric distribution. The imprecision model by Deng et al. implies a comparatively worse efficiency.

6.4.2.4. Comparison against HEX

HEX, which stands for Hierarchy and eXclusion, is a competing method for learning from imprecise data proposed in Deng et al. (2014). To the best of our knowledge, it is the only method next to CHILLAX capable of this feat. It relies on a larger set of assumptions, namely that in addition to subsumption (see section 5.2.2), there are concepts in the concept hierarchy that exclude one another. In their implementation, they assume that all concepts are mutually exclusive unless they share a common hyponym. CHILLAX does not make this assumption because it does not apply trivially to concept hierarchies that are DAGs. It is only straightforward to make for tree-shape concept hierarchies where no concepts have a common descendant. Otherwise, additional information is needed to determine whether two concepts that share a hyponym are mutually exclusive in a semantic sense or not (see section 5.2.2 for an example).

For a fair comparison, we choose hyperparameters such that the performance without any imprecision is comparable between CHILLAX and HEX. This setup isolates the effects of imprecision. The accuracies obtained by both baselines as well as HEX and CHILLAX without imprecision are shown in table 6.8. We also consider the top-5 accuracies (see section 6.1.1) as used in the ILSVRC2012 challenge (see section 6.2.1.1).

Introducing imprecision as defined by Deng et al., by replacing labels with their direct hypernyms with probability p, we obtain the results presented in table 6.9. With respect to the top-5 accuracy, CHILLAX outperforms HEX and both baselines in all scenarios. In terms of (top-1) accuracy, HEX produces a higher value of 41.50 % compared to 38.10 % of CHILLAX for $p = 0.99$. For $p = 0.95$ the difference is slightly smaller at 52.40 % and 52.10 %, respectively. In the remaining cases, CHILLAX performs best. Notably, the *leaves only* baseline reaches an accuracy of 60.20 % at $p = 0.5$, compared to 58.20 % of HEX. We further discuss these results in section 6.7.1.2.

Concise Results

CHILLAX vs. HEX: CHILLAX always performs better than HEX on ImageNet-1k in terms of top-5 accuracy, following the protocol by Deng et al. However, with respect to top-1 accuracy, HEX produces favorable results for very high imprecision, *i.e.*, $p \geq 0.95$.

6.4.3. Self-Supervised CHILLAX

In section 5.3.3, we argue that the “extrapolation” capability of CHILLAX, *i.e.*, generalizing from imprecise labels to precise predictions, should be applied during training as well. One reason is CHILLAX’s inability to learn from examples labeled as the root of the concept hierarchy as a result of its closed-world assumption. To apply extrapolation during training, we utilize pseudo-labels.

For a fair comparison, the experimental setup is identical to the previous experiments on CHILLAX, described in section 6.4.2.1. Here, we consider only the NABirds dataset, which allows for faster, and hence a larger number of, experiments com-

Table 6.8.: Large-scale benchmark comparison of CHILLAX, HEX and baselines on ImageNet-1k. We report the accuracy in percent on the validation set, with top-5 in parentheses. There is no inaccuracy and no imprecision. Table compiled of data from Brust, Barz, and Denzler (2021b).

Method	
Leaves only	**65.19** (**86.90**)
Random leaf	64.82 (86.46)
HEX	62.60 (84.30)
CHILLAX	62.52 (83.55)

Table 6.9.: Large-scale benchmark comparison of CHILLAX, HEX and baselines on ImageNet-1k. We report the accuracy in percent on the validation set, with top-5 in parentheses. There is imprecision as indicated, no inaccuracy. Table compiled of data from Brust, Barz, and Denzler (2021b) and Deng et al. (2014).

Deng et al. p	0.99	0.95	0.9	0.5
Leaves only	6.93 (16.45)	30.17 (51.58)	41.58 (64.03)	60.18 (82.25)
Random leaf	32.14 (68.10)	35.68 (71.76)	37.86 (74.22)	56.00 (81.87)
HEX	**41.50** (68.50)	**52.40** (77.20)	55.30 (79.40)	58.20 (80.80)
CHILLAX	38.08 (**68.55**)	52.13 (**78.07**)	**55.51** (**80.17**)	**62.14** (**83.58**)

pared to ImageNet-1k. However, since the self-supervised methods introduce further hyperparameters, we focus only on a representative subset of the noise models in an effort to improve readability. We select the following noise models, which shall be identified by their numeric indices henceforth:

(i) No noise, *i.e.*, no imprecision,

(ii) Relabeling to direct hypernym with $p = 0.99$ as proposed by Deng et al.,

(iii) Geometric distribution with $q = 0.5$,

(iv) Poisson distribution with $\lambda = 1$,

(v) Poisson distribution with $\lambda = 2$.

The respective fraction of precise labels can be found in table 6.3.

In this section, we offer an empirical evaluation of self-supervised CHILLAX focusing on three aspects. First, a simulation study to validate the feasibility of self-supervision on imprecise data and establish performance bounds. Second, an extensive comparison of the methods proposed in sections 5.3.3.2 to 5.3.3.4. And third, an analysis of the IC of the pseudo-labels compared to the original training data.

Table 6.10.: Hierarchical F1 score in percent between ground truth label in validation set and extrapolated imprecise label with imprecision model and extrapolation method as indicated, after learning from respective imprecise training data. Table compiled of data from Brust, Barz, and Denzler (2022).

Noise model	(i)	(ii)	(iii)	(iv)	(v)
Do nothing	100.00	87.49	59.79	61.73	78.06
Leaf node	100.00	**94.75**	81.15	79.12	93.20
d^* steps ($d^* = 1$)	100.00	94.74	73.97	74.81	87.27
d^* steps ($d^* = 2$)	100.00	94.74	79.29	79.03	91.31
d^* steps ($d^* = 3$)	100.00	94.72	80.88	79.29	92.91
Fixed threshold ($\theta = 0.55$)	100.00	90.99	**82.87**	**81.37**	**93.66**
Fixed threshold ($\theta = 0.8$)	100.00	90.72	81.19	80.03	92.99

6.4.3.1. Simulation of a Single Training Step

Self-supervised learning using pseudo-labels can introduce potentially harmful feedback loops, which we explain in detail in section 5.3.3.3. To estimate the accuracy of the generated pseudo-labels in an isolated manner, we consider only a single extrapolation step. First, we train one instance of CHILLAX on NABirds for each of the five noise models (i)-(v) using the respective modified training data. The modification process is detailed in section 6.4.1.1. Inaccuracy is not introduced.

We then modify the respective validation sets using the same noise models and apply the methods proposed in sections 5.3.3.2 and 5.3.3.3 to extrapolate predictions on the modified validation sets, independently for each noise model. Note that we cannot use the adaptive self-supervised method (see section 5.3.3.4) for this investigation since it depends on an internal state. As a “sanity check” baseline, we also compare to a “do nothing” method that performs no extrapolation at all.

Finally, we measure the hierarchical F1 score (hF1, see section 6.1.2) between the extrapolated predictions and the original labels in the unmodified validation set. Since the extrapolation results are not guaranteed to be in $\mathcal{Y}^P$, we cannot always determine the accuracy and have to rely on a hierarchical measure instead.

The results are presented in table 6.10. As a “sanity check”, we observe the results even when there is no imprecision (i). Because no method will extrapolate to concepts that are semantically incompatible with the ground truth, the resulting hF1 score should be exactly 100 %. This phenomenon is detailed in section 5.3.3.1. As a further consequence, it should be impossible for any method to produce a lower hF1 score than the control method, since the hierarchical recall cannot decrease. The results show that both assumptions hold. For noise model (iii), which represents the highest amount of imprecision in terms of IC (see table 6.3), the control method achieves an hF1 score of 59.79 %. The self-supervised methods range from 73.97 % up to 82.87 %. The *leaf node* method, which does not require any hyperparameter tuning, obtains 81.15 %. Compared to the control method, it recovers more than half of the hF1 score lost due to the imprecision. Models (iv) and (v) exhibit similar characteristics, with *fixed threshold* always performing best for at least one setting of θ.

However, the results for noise model (ii), which is the experimental protocol introduced by Deng et al., deviate. We already observe this trend in the previous experiment (section 6.4.2.2). Here, the hierarchy-informed methods (see section 5.3.3.3) clearly outperform the *fixed threshold* by more than three percent points. Still, all methods result in a substantial advantage compared to the control method, which we discuss in section 6.7.1.3.

Concise Results

Simulation: In terms of hierarchical F1, the hierarchy-informed and confidence-based self-supervised CHILLAX variants are able to recover more than half of the information lost to imprecision by extrapolating using predictions. The Deng et al. experimental protocol again produces qualitatively different results to our own noise models.

6.4.3.2. Large-Scale Results

In section 6.4.3.1, we establish that the methods proposed in sections 5.3.3.2 and 5.3.3.3 can recover more than half of the information, in terms of a hierarchical F1 score, that is lost from introducing imprecision (see section 6.4.1.1). The goal of this experiment is to observe the application of these methods in practice. Except for the self-supervision aspect, the setup is identical to section 6.4.2.1 without any inaccuracy, only imprecision. Each method is applied on top of CHILLAX, such that CHILLAX learns from labels extrapolated as proposed in section 5.3.3. Since the extrapolations are based on predictions by CHILLAX and accompanying confidence scores, a potentially harmful feedback loop occurs (see section 5.3.3.3). Furthermore, accuracy and hierarchical F1 score cannot be compared directly. Hence, the results from section 6.4.3.1 are likely not transferable.

Table 6.11 shows the results of our large-scale evaluation on NABirds for the methods in sections 5.3.3.2 and 5.3.3.3. Again, the "do nothing" control method corresponds to CHILLAX without any modifications. With no imprecision, the accuracy is 81.63 %, to provide a reference point. The method *leaf node* achieves higher accuracies than the control method in all cases, but only by a slim margin of at most 0.40 percent points. Moreover, there is no setting for which it is the best method. For *d^* steps*, we observe a strong dependence on an additional confidence threshold (see section 5.3.3.3). Without the threshold, it consistently performs worse than CHILLAX on its own. For $\theta = 0.9$, the results are competitive, and for the geometric noise distribution (iii), it achieves the best accuracy of 50.68 % compared to 49.04 % of CHILLAX.

The *fixed threshold* method is very competitive, but displays a remarkable dependence on its hyperparameter θ. It reaches the highest accuracies for imprecision models (ii), (iv) and (v) at $\theta = 0.994$, $\theta = 0.99$ and $\theta = 0.9$, respectively. A fixed *IC range* has negative effects on the accuracy compared to CHILLAX in all cases except for (ii), where it reaches an accuracy of 63.43 % compared to CHILLAX at 62.66 %.

Table 6.11.: Large-scale benchmark comparison of CHILLAX and non-adaptive self-supervised methods on NABirds. We report the accuracy in percent on the validation set. There is imprecision as indicated, no inaccuracy. Table compiled of data from Brust, Barz, and Denzler (2022).

Noise model	(ii)	(iii)	(iv)	(v)
Do nothing	62.66 ± 0.82	49.04 ± 1.04	43.18 ± 0.20	70.91 ± 0.34
Leaf node	63.05 ± 1.37	49.36 ± 0.48	43.49 ± 0.20	70.94 ± 0.42
d^* steps:				
$d^* = 1.0$	61.78 ± 0.27	33.11 ± 0.87	23.49 ± 0.60	65.44 ± 0.83
$d^* = 1.0$, $\theta = 0.8$	61.75 ± 0.69	48.09 ± 0.75	40.85 ± 1.26	71.67 ± 0.23
$d^* = 1.0$, $\theta = 0.9$	63.13 ± 0.70	49.98 ± 0.55	41.54 ± 1.37	71.75 ± 0.26
$d^* = 2.0$	61.31 ± 0.68	14.53 ± 0.82	12.12 ± 0.96	59.58 ± 0.60
$d^* = 2.0$, $\theta = 0.8$	62.07 ± 0.43	48.31 ± 0.84	37.21 ± 0.72	71.74 ± 0.81
$d^* = 2.0$, $\theta = 0.9$	62.52 ± 0.68	**50.68** ± 0.44	41.78 ± 0.47	71.54 ± 0.33
Fixed threshold:				
$\theta = 0.55$	61.48 ± 0.36	26.56 ± 0.94	22.68 ± 0.29	65.32 ± 0.53
$\theta = 0.8$	61.73 ± 0.54	39.80 ± 0.97	31.15 ± 1.65	69.86 ± 1.23
$\theta = 0.85$	61.93 ± 0.25	43.35 ± 0.72	34.60 ± 0.73	70.59 ± 0.17
$\theta = 0.9$	62.34 ± 0.33	46.74 ± 1.27	38.03 ± 0.78	**71.77** ± 0.00
$\theta = 0.95$	62.75 ± 0.21	48.66 ± 1.03	42.11 ± 1.48	71.47 ± 0.44
$\theta = 0.97$	63.00 ± 0.58	50.09 ± 0.26	43.20 ± 0.57	71.40 ± 0.25
$\theta = 0.99$	63.51 ± 0.52	49.37 ± 0.28	**44.02** ± 0.12	71.14 ± 0.15
$\theta = 0.992$	63.02 ± 0.57	48.88 ± 0.65	43.78 ± 0.33	71.21 ± 0.51
$\theta = 0.994$	**63.54** ± 0.51	49.23 ± 0.55	43.61 ± 0.94	70.99 ± 0.27
$\theta = 0.996$	63.11 ± 0.60	49.37 ± 0.61	43.69 ± 1.00	71.08 ± 0.45
$\theta = 0.998$	63.10 ± 0.83	49.16 ± 0.63	43.81 ± 0.25	70.96 ± 0.22
$\theta = 0.999$	62.78 ± 0.48	49.56 ± 1.04	42.92 ± 0.68	71.37 ± 0.39
IC range $[0.0, 0.2]$	62.11 ± 0.44	46.42 ± 0.71	40.50 ± 0.50	68.39 ± 0.47
IC range $[0.1, 0.3]$	61.78 ± 0.42	29.77 ± 0.98	26.07 ± 0.67	64.31 ± 1.05
IC range $[0.2, 0.4]$	63.43 ± 0.42	36.00 ± 1.29	30.64 ± 0.86	68.61 ± 0.22
IC range $[0.3, 0.5]$	63.23 ± 0.10	35.00 ± 0.46	31.45 ± 0.60	68.55 ± 0.94
IC range $[0.4, 0.6]$	62.96 ± 0.98	33.24 ± 0.65	27.31 ± 1.41	68.46 ± 0.33

Concise Results

Large scale: The hierarchy-informed and confidence-based self-supervised CHILLAX variants perform in different manners. *Fixed threshold* achieves the highest accuracies overall, but requires an infeasible amount of fine-tuning. *d^* steps* and *IC range* are only beneficial in some cases. *Leaf node* improves the accuracy over CHILLAX in all cases and is a simple heuristic without hyperparameters.

We present the detailed results of the continued large-scale evaluation in the appendix (table A.5), focusing on the adaptive method proposed in section 5.3.3.4. For comparison, we include CHILLAX as well as the best performing method from section 6.4.3.1 for each noise setting. Notably, in the case of (v), the Poisson distribution with $\lambda = 2$, two different settings ($\delta\mathfrak{I}^* = 0.0375$ and $\delta\mathfrak{I}^* = 0.5$), outperform the *fixed threshold*. With models (iii) and (iv), the best accuracy is lower than the highest performing respective non-adaptive extrapolation method, but still higher than the CHILLAX control method. However, following the experimental protocol of Deng et al. in (ii), we again observe opposite results, namely, that all settings of $\delta\mathfrak{I}^*$ result in worse accuracy compared to CHILLAX. With $\delta\mathfrak{I}^* = 0.025$, *adaptive threshold* obtains higher accuracies than CHILLAX by a very small margin, but over all imprecision models except for (ii). We discuss our findings in section 6.7.1.3.

Concise Results

Large scale: The adaptive self-supervised CHILLAX variant obtains consistent improvements over a wider range of its hyperparameter than *fixed threshold*. However, on average, *leaf node* performs better even without any parameterization.

6.4.3.3. Information Content Gain

Both the *adaptive threshold* and the *IC range* extrapolation methods rely on the assumption that a certain amount of IC can be gained "safely", *i.e.*, traded off with a reasonable fraction of inaccurate pseudo-labels. Regardless of the validity of this assumption, it is important to verify that the *realized* differences in IC are in close agreement to the specified values.

Constructively, the *adaptive threshold* method has no safeguards against individual examples deviating strongly from $\delta\mathfrak{I}^*$. Instead, it attempts to control its threshold such that the average IC difference matches $\delta\mathfrak{I}^*$ as closely as possible. *IC range* does not allow the IC difference to exceed the specified range's upper bound for any example. However, it cannot guarantee the lower bound in all cases, simply because the concept hierarchy might not allow it. For example, if the IC of an imprecise label is already 0.95, no extrapolation can result in an IC difference inside $[0.1, 0.3]$. Hence, the average IC difference can be biased towards lower values.

Table A.6 shows the realized IC differences during training, averaged over all optimization iterations. We first observe *adaptive threshold*. While there is no clearly visible bias towards either direction, the realized IC differences are on average higher than the respective value of $\delta\mathfrak{I}^*$ for $\delta\mathfrak{I}^* \leq 0.3$ and lower for $\delta\mathfrak{I}^* > 0.3$. The imprecise labels of imprecision model (ii) have an average IC of 0.83 (see table 6.3), which leads to an upper limit of 0.14 as the average precise IC is 0.97. Hence, neither method can actually realize a higher IC gain. *IC range* on average shows results that are biased towards lower values. It is not able to enforce a lower limit of 0.30 or higher in any case. The results are discussed further in section 6.7.1.3.

Concise Results

IC gain: The *adaptive threshold* method can successfully control the IC gain, keeping it close to its hyperparameter $\delta\mathfrak{I}^*$. *IC range* exhibits a clear bias towards lower realized IC gains.

6.5. Imprecise Data from Webly Supervision: Moths

In this section, we present a case study for the utility of imprecise data, in the form of images crawled from the web. The term *webly supervised learning*, similar to weakly supervised learning, is used to indicate that the supervision signal has a different quality (see also section 4.2.4). We consider a very specific problem domain, where no annotated data is available publicly.

The problem concerns the biodiversity research project AMMOD[9] and specifically moths. In order to determine seasonal, meteorological and other trends in species abundance, activity and behavior, the project installs “moth scanners”. An illuminated canvas screen is filmed by a camera, the *SpeciesMothCam*, in regular intervals.

This filming results in a dataset, which at the time of writing is not published separately. However, it is discussed in Böhlke et al. (2021) and available for use in this thesis. It is a classification dataset which requires distinguishing between 200 different species of methods. There are 2205 images in total, which are split into two parts with 100 disjoint species each. The parts are denoted “B1” and “B2”. Images can contain many moths at a time, however crops of a single representative moth per image are available, which are guaranteed to fit the single annotation.

6.5.1. Setup

This study involves several steps, each further processing the moth dataset. First, we acquire a taxonomy from WikiSpecies, which is detailed in section 3.3.3.1, to construct a concept hierarchy. We then compare the accuracy of our hierarchical classifier (see section 5.2) to an OHSM baseline to measure the effect of integrating domain knowledge. The technical setup of the hierarchical classifiers and the baseline match the setup used for NABirds in section 6.4.2.1. However, we reduce the number of optimization iterations to 9024 for the combined dataset and 4512 for B1 and B2 individually.

---

[9] `https://www.ammod.de/` (last accessed July 5th, 2021) and Vogel (2017)

Second, we download images from Flickr by using each concept in the concept hierarchy as a search term in the database. We then apply CHILLAX (see section 5.3.2) to the resulting imprecise data.

Third, we filter the downloaded images using an unsupervised approach leveraging our hierarchical classifier. The predicted probability of the root concept is used to construct an ordering of the images without regard for the possibly incorrect label.

6.5.1.1. Mapping WikiSpecies to the Dataset

Initially, the moth dataset ships with a set of 200 mutually exclusive species. To acquire a matching concept hierarchy, we use the WikiSpecies knowledge base (section 3.3.3.1).

First, the species names used in the dataset have to be mapped to the respective taxa as stored in WikiSpecies. In general cases, mapping concepts is highly non-trivial, which we discuss further in section 5.1.2. However, the species names used in this biological application are not as flexible as regular words and rarely used in derivative forms. Thus, an automated mapping of species to taxa should be reliable enough for a majority of concepts in the dataset. WikiSpecies contains direct matches for 191 of the 200 species. We map the remaining 9 species manually, as specified in table A.7. Of these species, one is a typing mistake. Six are groups of species, which we map to their respective genera unless those would overlap other species in the dataset. In these cases, we create new concepts under the genera. Two are missing from WikiSpecies.

After the mapping, we build the concept hierarchy by recursively exploring the supertaxa (hypernyms) of each taxon (concept) in WikiSpecies. We then remove redundant information in the concept hierarchy, *i.e.*, concepts with exactly one hypernym and one hyponym, while called *monotypic taxa*. The root is represented by the species *Arthropoda*, and in total the concept hierarchy contains 275 concepts.

6.5.1.2. Downloading Data from Flickr

With the complete concept hierarchy, we now submit each concept as a search term in the Flickr online photo sharing community (see also section 6.6.3). Figure A.1 displays the average number of search results per concept returned by Flickr, in relation to the concept's depth in the hierarchy. It resembles a very rough approximation of a geometric distribution (see also sections 5.1.1.1 and 6.6.3).

The search, dated January 7th, 2021, produces 1 454 743 results in total, with an average of 5290 results per concept. From these results, we download up to 75 images per concept, sorting the results by number of downloads as a coarse indicator of quality. Since not all searches lead to enough results, only 18 225 of the possible 20 625 images are downloaded. To avoid confusion, we remove images that occur as results to more than one search term, leaving 16 777 remaining images. Of these, 12 918 are annotated as one of the 200 leaf nodes representing the original moth species.

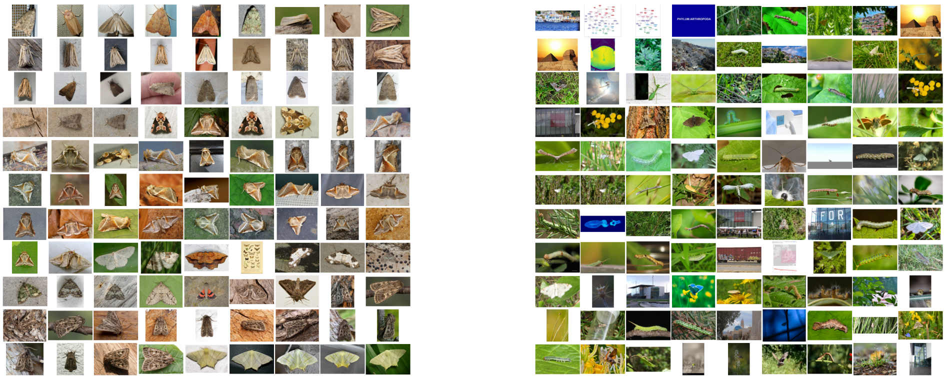


Figure 6.2.: Output of unsupervised filtering process on Flickr moths. Left: highest-scoring images. Right: lowest-scoring images. All images downloaded from Flickr.

6.5.1.3. Unsupervised Filtering

While images on Flickr are annotated manually by humans, its users strive for different objectives from paid dataset annotators. We discuss the users' goals in section 6.7.3.3. Furthermore, the users are not presented with images to annotate, but submit them themselves. As such, there is no controlled domain of image data in Flickr, and it is certainly not restricted to moths.

For example, a short exploratory qualitative analysis reveals that certain taxa are also used as female first names. This results in a number of portraits in the downloaded images. Moreover, the data is not balanced because of the varying number of search results per concept.

Hence, we propose to filter the images. We apply the trained hierarchical classifier obtained from learning the original moth dataset as described in section 6.5.2.1 in its combined and cropped form. The predicted probability for the root of the concept hierarchy, *Arthropoda*, is interpreted as the probability of any image belonging to the domain of moths. This test of the closed-world assumption (see section 5.2.2) effectively performs novelty detection (see section 7.1.1). For reference, we show qualitative examples of extreme probabilities in fig. 6.2.

Note that the filtering process does not use the labels of the Flickr data. It is an unsupervised process. We keep the 5000 candidates with the highest respective probabilities for *Arthropoda*. Of these images, 4339 are labeled as one of the 200 leaf node, which is a substantially higher fraction than in the unfiltered images.

Table 6.12.: Accuracy in percent, on validation sets of respective moth dataset part.

Dataset	HC	Baseline
Cropped to single moth:		
B1	**95.00** ± 0.94	94.50 ± 1.27
B2	**92.74** ± 1.09	92.54 ± 1.27
Both	**92.62** ± 0.72	92.07 ± 0.71
Uncropped:		
B1	76.50 ± 2.76	**81.90** ± 1.75
B2	79.40 ± 1.03	**83.68** ± 1.87
Both	70.47 ± 2.87	**79.65** ± 1.86

> We exploit an implementation detail of the classifier in section 5.2, where $\mathbb{P}(\tilde{y}_{\text{root}}{}^{+})$ is not fixed to 1, but learned from the training data. While all examples are labeled as 1 w.r.t. the root, the predicted probability varies slightly.

6.5.2. Results

We analyze the individual results of this experiment in the same order as the steps are proposed in section 6.5.1. First, we compare the performance of our hierarchical classifier against the OHSM baseline to estimate the value of the integrated domain knowledge from WikiSpecies. Second, we add the (imprecise) data downloaded from Flickr and measure the performance of CHILLAX and the baseline. Third, we study the effects of our unsupervised filtering setup. A discussion of the results is offered in section 6.7.2.

6.5.2.1. Integrating Domain Knowledge

Table 6.12 shows the results of the initial experiment, which compares the hierarchical classifier (see section 5.2) against the OHSM baseline. When analyzing the cropped variants of the datasets, the hierarchical classifier reaches a higher accuracy than the baseline on all combinations. For the combined dataset, it obtains an accuracy of 92.62 % compared to 92.07 % of the baseline. However, concerning the uncropped datasets, we observe the opposite relation. For example, the baseline reaches 79.65 % on the uncropped dataset combining both parts, while the hierarchical classifier shows only 70.47 %.

In the following experiment, we consider only the cropped variant of the combined datasets. Crops are more likely to be used in a practical application involving moth classification, as opposed to detection where the uncropped variant would be more interesting.

Table 6.13.: Accuracy in percent on combined moth validation set. The training data is enriched as indicated.

Dataset	CHILLAX	Baseline	Böhlke et al. (2021)
Original data only	**92.62** ± 0.72	92.07 ± 0.71	72.75 ± 1.46
Unfiltered Flickr images:			
All taxa	**96.66** ± 0.45	96.26 ± 0.53	
Leaf nodes only	97.21 ± 0.33	**97.46** ± 0.21	95.43 ± 0.57
Filtered Flickr images:			
All taxa	96.56 ± 0.27	**96.66** ± 0.55	
Leaf nodes only	97.06 ± 0.76	**97.76** ± 0.31	95.93 ± 0.45

6.5.2.2. Adding Webly Supervision

After studying the effects of integrating the taxonomic knowledge from WikiSpecies, we now add the images downloaded from Flickr. Since many of the labels are imprecise, we replace our hierarchical classifier with CHILLAX (see section 5.3.2). The OHSM baseline uses the *random leaf* strategy as detailed in section 6.4.2.1 to process imprecise data. The results of this experiment are presented in Table 6.13 (see the "unfiltered" section).

Overall, incorporating the Flickr data increases the accuracies of both methods on the moth validation set substantially, by approximately four percent points. Utilizing the full Flickr data, CHILLAX obtains an accuracy of 96.66 % compared to the baseline at 96.26 %. In Böhlke et al. (2021), a similar experiment is performed on part B1 of the dataset. With their approach to web crawling and filtering, the authors obtain an accuracy of 95.93 % on half of the validation set we use, and only 100 species. Their filtered dataset consists of 9424 images crawled from the web, including 300 seed images from the original training data.

To gauge the "value" of the imprecise data on its own, we also analyze a situation where only the images labeled as leaf nodes are processed. Here, the baseline performs marginally better than CHILLAX. While CHILLAX cannot leverage imprecise data in this case, it should still have the advantage of integrated domain knowledge. Either way, the precise Flickr dataset produces better performance than its imprecise counterpart.

Concise Results

Flickr data: Combining the moth dataset with data downloaded from Flickr increases accuracy from 92.62 % to 96.66 % for CHILLAX, which outperforms the baseline in both cases. However, learning from only the precisely labeled part results in higher accuracies for both methods, and a slight advantage for the baseline.

6.5.2.3. Unsupervised Filtering

We finally apply the unsupervised filtering scheme proposed in section 6.5.1.3. The results of this process are shown together with the previous experiment's results in table 6.13. The filtering step harms the accuracy of CHILLAX, but improves the accuracy of the baseline in each case, imprecise or precise. For CHILLAX, the filtering leads to a decrease in accuracy by around 0.10 of a percent point. At the same time, the performance of the baseline increases by approximately 0.40 percent points, reaching a maximum of 97.76 %.

As a consequence, the overall best performing setup requires the following unintuitive steps:

1. Train a hierarchical classifier or CHILLAX (equal for precise data) on the cropped, combined moth dataset,
2. use this classifier to predict probabilities which inform the filtering process,
3. train an OHSM classifier on the moth dataset, combined with the Flickr data which is filtered using CHILLAX, but use only the precise examples in each.

Ultimately, the highest performance is achieved by the baseline, but the processing steps leading to this result require a hierarchical method. We discuss our findings further in section 6.7.2.

Concise Results

Filtering Flickr data: Using CHILLAX to filter the downloaded images for outliers improves the performance of the baseline, but has a marginal negative effect on the accuracy of CHILLAX. Overall, the Flickr images can be used together with CHILLAX as a filter to increase the accuracy from 92.07 % to 97.76 %.

6.6. Pitfalls and Dangers

The methods and models proposed in chapter 5 introduce a number of highly non-trivial assumptions. In this section, we perform experiments to verify these assumptions, specifically concerning the correspondence between the visual and semantic domains, the correctness of concept hierarchies and the validity of depth-based modeling of imprecision (see section 5.1.1.1). Furthermore, we analyze the potential consequences should the assumptions not hold.

6.6.1. Visual-Semantic Correspondence

Partial results of the work presented in this section are published in Brust and Denzler (2019b).

Images and semantics are unalike modalities, whose differences are analyzed in Collell Talleda (2016) and Deselaers and Ferrari (2011). This domain gap, related to the semantic gap discussed in Barz and Denzler (2021a), can present a hurdle for integrating domain knowledge in cases where the problem domain and the knowledge domain are different. For example, the hierarchical classifier described in section 5.2 leverages semantic knowledge, *i.e.*, concept hierarchies, to solve a visual problem, *i.e.*, image classification.

It is implicitly assumed that relations between concepts have a counterpart in relations between images. Hence, some visual features should exist, which are shared by all concepts subsumed by the same hypernym, but no other concepts (see section 5.2.2). Other methods make the assumption explicit, *e.g.*, embeddings (see section 4.1.2) that optimize distances in a visual feature space to match semantic distances.

However, such a perfect correspondence is unlikely, and it depends strongly on the specific choice of visual features. For example, consider an image classification task. There are many possible images of the same object, *i.e.*, with the same semantics. There can also be many visually different objects within a single concept. Hence, images have many degrees of freedom w.r.t. their semantics. These are commonly removed by selecting an appropriately invariant feature representation (see section 2.3.4.2). Consequently, a feature representation where distances correspond perfectly with semantic distances has to map all images of a concept to a single point. Such a feature representation is already a perfect classifier, which is unlikely to exist.

Furthermore, there are arguments in favor of a “natural” visual-semantic correspondence, which is discovered by CNNs. Confusions of such a network correlate with semantic distances, even if the network does not integrate a concept hierarchy (see Bilal et al. (2018) and section 4.1.5). In the following, we explore the correspondence by observing Spearman correlations between different measures of similarity in the visual and semantic domains. The experiment compares pairwise similarities over all images and all concepts in the CIFAR-100 dataset (see section 6.2.1.1).

6.6.1.1. Measures of Semantic Similarity

In section 6.1.2.2, we discuss a small selection of measures of semantic similarity between concepts. For this experiment, we use five different formulations, which are identified as S1-S5, to obtain as general a sense of semantic similarity as possible. The measures, which are all computed w.r.t. the WordNet concept hierarchy (see section 3.3.1.1), are as follows:

S1 Graph distance, or Rada distance d_G, see eq. (6.2),

S2 maximum-depth bounded similarity (Resnik 1995, p. 3),

S3 intersection over union of the sets $\tilde{y}$, see eq. (5.2) and Maedche and Staab (2001, p. 4),

S4 distinct to shared $\tilde{y}$ feature ratio, see Sánchez et al. (2012, p. 7723), and

S5 IC-based distance (Jiang and Conrath 1997, p. 8) using eq. (5.1).

6.6.1.2. Measures of Visual Similarity

For visual similarity, we again select five different measures to obtain a diverse and representative sample. Here, our goal is to capture varying levels of abstraction, from pixels to complex features learned by CNNs. The following definitions are used in our experiment, where distances are inverted into similarities as shown in eq. (6.3):

V1 Mean squared difference,

V2 mean absolute difference,

V3 structural similarity index (SSIM) proposed in Wang et al. (2004),

V4 euclidean distance between GIST descriptors (*cf*. Oliva and Torralba 2001), and

V5 confusions of five OHSM classifiers trained on CIFAR-100 (see section 6.3.1).

We compute V1-V4 on all pairs of images, and then derive pairwise concept similarities by computing the means over the respective concepts. V5 directly gives pairwise similarities over concepts through its confusion matrix.

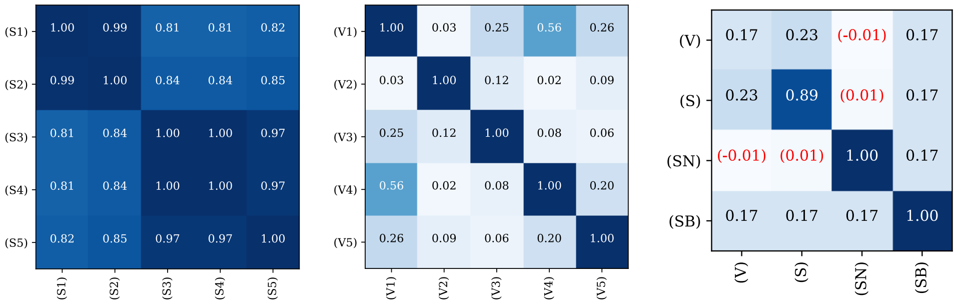
6.6.1.3. Results

We first observe the correlations within the two groups of similarity measures, to determine their mutual agreement. The semantic similarity measures agree with one another with a high rank-correlation of 0.89 on average, where we exclude the correlations between identical methods. Figure 6.3a shows the detailed results within the semantic group. Our methods of measuring visual similarity differ more strongly, with an inner correlation of only 0.17, suggesting a more diverse measurement. Correlations between the individual methods are presented in fig. 6.3b.

To bridge the domain gap, we aggregate each group (visual and semantic) by normalizing the output of the individual methods and computing the average visual and semantic similarities between concepts. Figure 6.3c shows the correlations between the two groups of similarity measures resulting from the aggregation. In this setting, the correlation between visual and semantic similarity on CIFAR-100 is 0.23.

Moreover, we add a “semantic baseline”, a similarity measure which is one for identical concepts and zero otherwise. This represents the knowledge integrated into an OHSM classifier, namely that visual features of distinct concepts do not overlap (*cf*. Niemann 1983). The correlation between this baseline and the aggregated visual similarity measures is only 0.17, compared to 0.23 of the actual semantic similarities based on the concept hierarchy. As a sanity check, we add a random similarity measure called “semantic noise” (SN), which is, as expected, not correlated with any measure but itself.

While the correlation is not particularly high, it shows that semantic similarity and visual similarity are related on a higher-than-trivial level, such that the concept hierarchy contains true knowledge about visual features that would not be available to a classifier otherwise. For a qualitative impression on CIFAR-100, we provide fig. 6.4, which shows pairs of concepts in the dataset where the visual and semantic similarity measures agree or disagree strongly. The findings are discussed further in section 6.7.3.1. We also construct a synthetic dataset where this correlation can be controlled in section 6.6.2.2.



	(S1)	(S2)	(S3)	(S4)	(S5)
(S1)	1.00	0.99	0.81	0.81	0.82
(S2)	0.99	1.00	0.84	0.84	0.85
(S3)	0.81	0.84	1.00	1.00	0.97
(S4)	0.81	0.84	1.00	1.00	0.97
(S5)	0.82	0.85	0.97	0.97	1.00

(a) Semantic similarities

	(V1)	(V2)	(V3)	(V4)	(V5)
(V1)	1.00	0.03	0.25	0.56	0.26
(V2)	0.03	1.00	0.12	0.02	0.09
(V3)	0.25	0.12	1.00	0.08	0.06
(V4)	0.56	0.02	0.08	1.00	0.20
(V5)	0.26	0.09	0.06	0.20	1.00

(b) Visual similarities

	(V)	(S)	(SN)	(SB)
(V)	0.17	0.23	(-0.01)	0.17
(S)	0.23	0.89	(0.01)	0.17
(SN)	(-0.01)	(0.01)	1.00	0.17
(SB)	0.17	0.17	0.17	1.00

(c) Groups of similarities

Figure 6.3.: Matrix of rank correlation coefficients between different measures of similarity, grouped by semantic and visual domain. Significance of $p < 0.05$ except for numbers in parentheses. Main diagonal represents inter-agreement R within the group. Legend: (V) — visual, (S) — semantic, (SN) — semantic noise, (SB) — semantic baseline. Figure taken from Brust and Denzler (2019b).

Concise Results

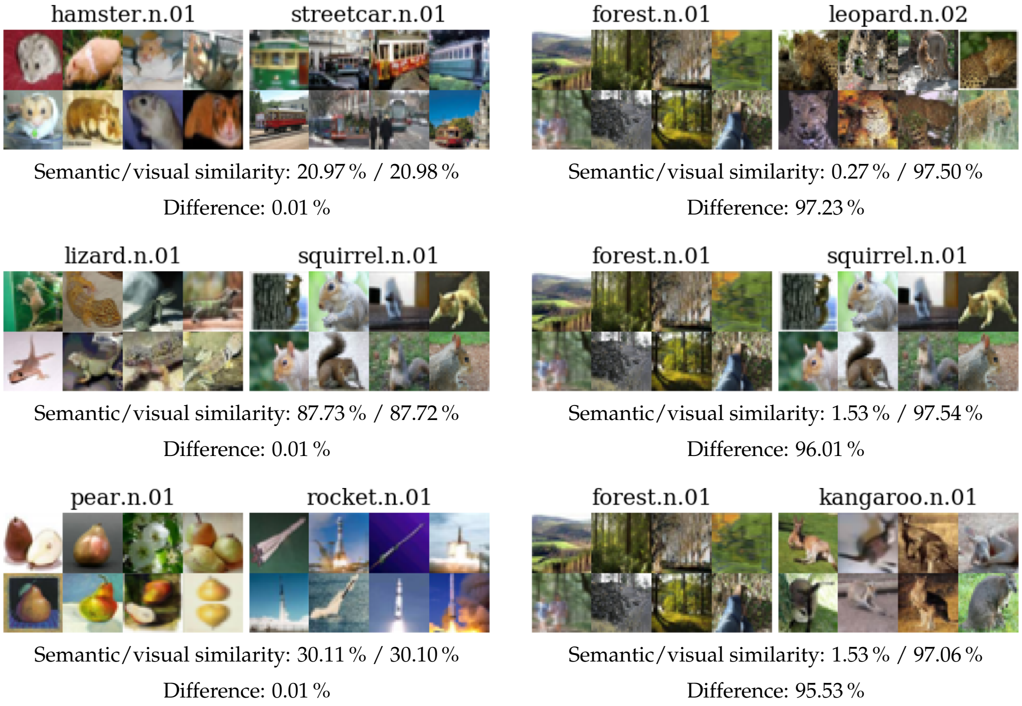
Visual and semantic similarity are (rank-)correlated with a coefficient of $R = 0.23$, when WordNet is used as a concept hierarchy. The correlation between the semantic baseline, which considers all distinct concepts equally dissimilar, and visual similarity is only $R = 0.17$.

6.6.2. Harmful Hierarchies

In the previous experiment, we investigate the discrepancies between the visual and semantic domains (section 6.6.1), which may lead a hierarchical classifier to faulty assumptions (see section 5.2.2), and ultimately, mispredictions. While a weak correlation between visual and semantic similarity is already cause for concern, there is one possibly more immediate contributor to mispredictions. The concept hierarchy itself could be "wrong", *i.e.*, the hyponymy relation could contain elements that do not correspond to the real world.

It should be noted that not all concept hierarchies have a true state that can be determined by objective measurements. For example, there is no widespread agreement on whether the `Renault Twizy` is a car, and even the legal definition has changed over time. Still, there exist relations that are grounded objectively, such as some (but not all!) biological taxonomies that rely on genetics (see section 3.3.3.1). A false measurement or data entry error could then result in a concept hierarchy that is objectively wrong, *i.e.*, constitutes false domain knowledge.

In the following, we propose a probabilistic model of such errors, and empirically evaluate their effects on a hierarchical classifier (see section 5.2) on the NABirds dataset. To isolate the effect from potentially already existing visual-semantic gaps, we then describe a synthetic dataset with perfect visual-semantic correspondence on


hamster.n.01
streetcar.n.01
Semantic/visual similarity: 20.97 % / 20.98 %
Difference: 0.01 %
lizard.n.01
squirrel.n.01
Semantic/visual similarity: 87.73 % / 87.72 %
Difference: 0.01 %
pear.n.01
rocket.n.01
Semantic/visual similarity: 30.11 % / 30.10 %
Difference: 0.01 %
forest.n.01
leopard.n.02
Semantic/visual similarity: 0.27 % / 97.50 %
Difference: 97.23 %
forest.n.01
squirrel.n.01
Semantic/visual similarity: 1.53 % / 97.54 %
Difference: 96.01 %
forest.n.01
kangaroo.n.01
Semantic/visual similarity: 1.53 % / 97.06 %
Difference: 95.53 %

(a) Highest agreement

(b) Lowest agreement

Figure 6.4.: Concepts in CIFAR-100 with respective highest and lowest ranking agreement between aggregated visual and semantic similarity measures (S) and (V). Figure taken from Brust and Denzler (2019b).

which we repeat the experiment. We compare the hierarchical classifier against an OHSM baseline. Furthermore, we consider a concept hierarchy where all inner nodes are removed to estimate the importance of the mutual exclusivity assumption made by an OHSM classifier.

6.6.2.1. Parent Replacement

For simplicity, we consider tree hierarchies as opposed to DAGs (see section 3.2.2.2) and assume only one type of error, which we denote as *parent replacement*. In a parent replacement, a concept c is assigned an incorrect parent, *i.e.*, hypernym, randomly chosen out of the parents of all concepts at the same depth as c (see section 5.1.1.1). Starting from the deepest level in the hierarchy k_{max}, where the precise concepts are located, we perform a parent replacement for each concept at depth k with a probability p_k defined as:

$$p_k = q^{k_{\text{max}}-k} \cdot (1-q) \cdot p_{\text{replace}} \,. \tag{6.4}$$

We scale the overall effect in our experiments by p_{replace}, such that $p_{\text{replace}} = 0$ represents no modifications. q parameterizes a geometric distribution that determines the probability of replacement depending on the depth. We set $q = 0.2$, resulting in $p_k = 0.8$ for the precise concepts, 0.16 for the level above, etc., for $p_{\text{replace}} = 1$.

We choose a geometric distribution as it encodes our expectation that mistakes are much more likely in higher depths, *i.e.*, closer to the leaf nodes than the root of the concept hierarchy. The motivation follows a similar reasoning to the volunteer noise model described in section 5.1.1.1. We assume that mistakes are easier to make and to notice when they are less “groundbreaking”. For example, assigning a subspecies of bird to the incorrect species is more likely to go unnoticed than claiming that a `songbird` is an `insect`.

It should be noted that p_{replace} is somewhat unintuitive in that 1.00 is not the maximum, *i.e.*, there remains some information in the concept hierarchy, even if only very little. This is a result of the finite depth k, such that the sum over all p_k does not reach its limit of 1.00.

6.6.2.2. D-CHIVES Synthetic Dataset

In the introduction of this experiment’s section, we determine two possible contributors to mispredictions of hierarchical classifiers. The first is the weak correlation between visual and semantic similarity (see section 6.6.1), and the second is a concept hierarchy that contains false information.

While the experiment aims to study the effects of the latter, the mispredictions due to the former cannot be separated when using a benchmark dataset consisting of natural images. To separate the individual effects, we propose a new dataset: D-CHIVES (**d**ataset and **c**oncept **h**ierarchy of **i**mages with **v**isuals **e**qual to **s**emantics).

We first generate a concept hierarchy. To determine the extent, we specify the total number of levels as well as upper and lower bounds for the *fan-out*. Starting

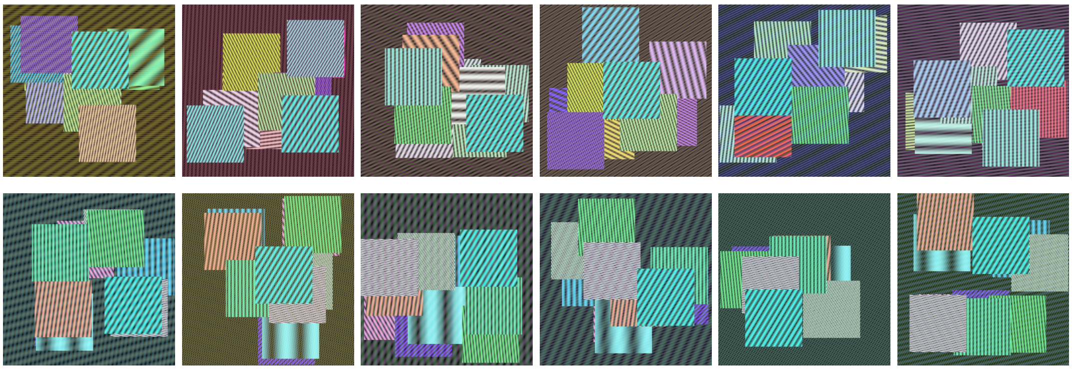


Figure 6.5.: Example images of the D-CHIVES L dataset. The top row shows images from different classes, while the bottom row shows images from the same class.

with a single root, a new level is generated by adding a random number of children within the specified fan-out bounds to each (current) leaf node. Concept names are determined randomly by combining pronounceable syllables from a list. We then generate a fixed amount of individual images.

The image synthesis starts by sampling a random precise concept from the concept hierarchy as its label. We then compute a feature vector indicating the presence of each concept in the hierarchy based on the label encoding in section 5.2.4.1. It is one for the label concept and all its hypernyms up to the root, and zero otherwise. This feature vector is translated into an image of arbitrary sizes in three steps.

First, the image is initialized with zeros. Second, for each positive entry in the feature vector, we place a small patch in the image at a random location. It contains sine waves of random phase and amplitude, which are identical for all instances of a patch representing the same concept's presence. Third, random sine waves are added on top of the image such that "solving" the dataset still requires a certain amount of generalization, as opposed to memorization.

For our experiment, we use "D-CHIVES L", which consists of 50 000 images, 256×256, pixels in size with an equal training and validation split (see section 2.2.2.1). The concept hierarchy contains 4927 concepts, 2971 of which are precise. Figure 6.5 shows visual examples from the dataset.

6.6.2.3. Results

The results of our experiment are presented in table 6.14 for both datasets. We initially focus on NABirds, where several observations of interest can be made.

First, the level-wise parent replacement operation described in section 6.6.2.1 decreases accuracy on average, as expected. While the hierarchical classifier reaches an accuracy of 81.66 % with the original hierarchy, it decreases to 78.89 % when $p_{\text{replace}} = 1$. Second, the effect of the level-wise parent replacement procedure on accuracy is more pronounced for $0.0 \leq p_{\text{replace}} \leq 0.6$ than it is for the higher settings. Third, similar to section 6.3.2.2, the OHSM baseline is superior to all settings of the hierarchical classifier, even if the hierarchy is not modified at all. And finally, there is

Table 6.14.: Accuracy (%) of hierarchical classifier and baseline on NABirds and D-CHIVES L validation sets. Hierarchy modified as indicated.

Method / Dataset	NABirds	D-CHIVES L
Baseline		
No Hierarchy	**82.78** ± 0.07	76.02 ± 0.80
Hierarchical Classifier		
Flat Hierarchy	1.47 ± 0.45	0.36 ± 0.16
Unmodified Hierarchy	81.66 ± 0.24	**88.98** ± 0.07
$p_{\text{replace}} = 0.2$	80.50 ± 0.23	77.39 ± 0.21
$p_{\text{replace}} = 0.4$	79.57 ± 0.09	65.38 ± 0.44
$p_{\text{replace}} = 0.6$	79.35 ± 0.06	55.97 ± 0.88
$p_{\text{replace}} = 0.8$	79.44 ± 0.08	47.90 ± 0.99
$p_{\text{replace}} = 1.0$	78.89 ± 0.08	40.96 ± 0.33

a complete failure of the hierarchical classifier when the inner nodes of the concept hierarchy are removed, resulting in an accuracy of 1.47 %.

To isolate the effects and offer more robust conclusions to the observations in the previous section, we repeat the experiment on D-CHIVES L. On this dataset with perfect visual-semantic correspondence, we make the following observations.

Again, accuracy decreases as p_{replace} increases, confirming the first observation of the NABirds experiment. However, the effect size is substantially larger. The effect of the level-wise parent replacement procedure on accuracy is still more pronounced for lower p_{replace}, but only very slightly. In fact, it is almost linear ($R = -0.9443, p < 0.05$). On D-CHIVES L, the OHSM baseline only has comparable accuracy to the hierarchical classifier for $p_{\text{replace}} \geq 0.2$. With an unmodified hierarchy, the hierarchical classifier strongly outperforms the baseline by 12.96 percent points. Finally, the hierarchical classifier fails again when the concept hierarchy is stripped of inner nodes.

We discuss the aforementioned findings in section 6.7.3.2.

Concise Results

Harmful Hierarchies: Introducing faulty elements to a concept hierarchy has an adverse effect on accuracy. The effect scales with visual-semantic correspondence. On NABirds, less than three percent points are lost, while on D-CHIVES L, accuracy decreases from 88.98 % to 40.96 %.

6.6.3. Imprecision in Real-World Data

Partial results of the work presented in this section are published in Brust, Barz, and Denzler (2021b).

Statistical models of imprecise data are important for benchmark purposes because they allow for experiments where original, precise labels are known and can be compared with. However, these models make strong assumptions about the unique

Title “Impressive yachts in the quaint harbor of Korčula”
Description “Croatia is a country in Southeast Europe. It borders Slovenia to the northwest, Hungary to the northeast, Serbia to the east […]”
Tags trees islands Croatia Kroatië coastline coast Hrvatska nature island Korčula Korcula…

Figure 6.6.: Example photograph and metadata from Flickr. Image captured from `https://flic.kr/p/2jwRjCc` on March 2nd, 2021.

properties of data sources. Hence, any results from experiments using them can only generalize to real-world scenarios if the assumptions are correct. In this section, we present a study to validate the models described in section 5.1.1.1.

We build a dataset from 1 500 000 photographs which were posted to Flickr[10] between January 1st and December 31st, 2019. All images are accompanied by metadata, *i.e.*, a title, a short textual description, and a set of user-specified tags (see fig. 6.6 for an example). There is also machine-generated metadata available which we ignore for the purposes of this study as it is only based on the information supplied by the user. We attempt to map each photograph to a single WordNet (*cf.* Miller 1995, also section 3.3.1.1) synset using the metadata and the algorithms described in section 5.1.2. Each type of metadata has a different purpose from a user’s perspective, so we investigate synsets extracted from title, description, and tags separately.

The first processing step is *tokenization*. Title and description are strings and require splitting into individual words for further analysis. This step also removes spaces and punctuation. We use NLTK’s `TweetTokenizer` as it is aware of platform- and community-specific language phenomena such as emoticons and hashtags. The resulting collection of words is considered a set, and we ignore the order.

Afterwards, the next step is lemmatization. We use `MORPHY` (Beckwith and Miller 1990, see also section 5.1.2.1) on each word individually to obtain lemmata. If the part-of-speech of a word cannot be determined uniquely, we assume nouns — trading off potential misclassifications for a greater selection of synsets later. Unlemmatized words that match lemmata in WordNet exactly are associated with the respective synsets. The remainder is matched after lemmatization. If multiple synsets have lemmata that match one extracted lemma exactly, we select the least specific synset by depth (see section 5.1.1.1). We finally use the *most* specific synset from the results

---

[10] `https://www.flickr.com/`

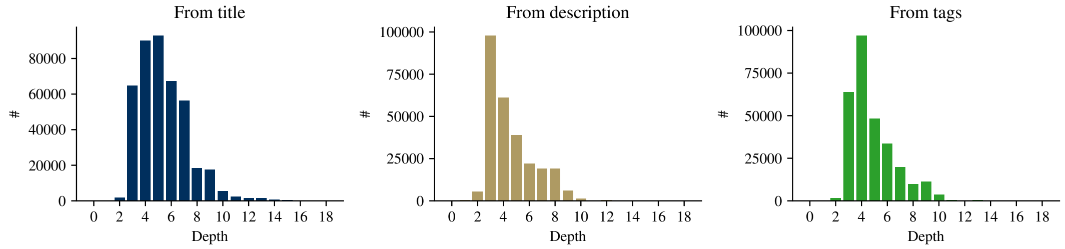


Figure 6.7.: Frequency of synsets at depth (distance from root) in WordNet hierarchy. Synsets extracted separately from title, description and tags of Flickr images. Figure taken from Brust, Barz, and Denzler (2021b).

to represent the title, description or taglist.

Figure 6.7 shows the frequency of each possible depth in the WordNet hierarchy occupied by synsets extracted from photographs' metadata. Analyzing the image title, the most common depth is 6. Typical synsets at the depth are, *e.g.*, `celebration.n.01`^{W}, `boulder.n.01`^{W}, and `sunrise.n.03`^{W}. On average, a synset extracted from the title is at depth 5.38 and the overall frequencies resemble a Poisson distribution. The synsets extracted from the textual description appear as a shifted or offset geometric distribution. Here, the mode is 3 with an average depth of 4.56. One can expect general synsets such as `series.n.06`^{W},`image.n.07`^{W}, or `head.n.17`^{W}. Finally, the depth distribution of synsets that are taken from the set of tags is a mixture of the aforementioned title and textual description distributions. The most common depth at which a synset occurs is in between, at 4, and the average depth is 4.82. Here we encounter slightly more specific synsets. Examples are `night.n.04`^{W}, `party.n.05`^{W}, and `west.n.08`^{W}.

We discuss the findings in more detail in section 6.7.3.3.

Concise Results

Imprecision in Flickr: Through the lens of the WordNet hierarchy, we observe both the Poisson and geometric distributions over precision in the Flickr metadata. The distributions appear "shifted" towards higher depths.

6.7. Summary and Discussion

This section summarizes our findings from the previous sections. We offer conclusions, discuss the results and propose further hypotheses. To this end, we interleave experimental evidence and commentary, but provide a visual distinction by formatting. Moreover, we explain the omissions and limitations of this work concerning the selection of datasets, baselines and evaluations performed.

6.7.1. Benchmark Datasets

We first consider results on benchmark datasets, where we separate the two use cases of the hierarchical classifier as discussed in section 5.2. Results regarding knowledge integration are detailed in section 6.3, whereas experiments on imprecise data are performed in section 6.4.

6.7.1.1. Knowledge Integration

In the small-scale setting, we find that the hierarchical classifier requires fewer training steps to reach a certain accuracy than the OHSM baseline. It also outperforms the baseline on two out of three datasets. At large scale, the OHSM baseline outperforms the hierarchical classifier in all settings, but only marginally. We speculate that the bad performance of small-scale models compared to their large-scale counterparts is due to two main reasons. First, the small-scale models produce substantially more errors, which creates more possibilities for the assumptions in section 5.2.2 to correct potential mistakes.

Second, the hyperparameters and initialization for the large-scale models, at least on NABirds, originate from state-of-the-art models (Cui et al. 2018). As such, are likely fine-tuned through extensive grid search, whereas the hyperparameters for our hierarchical classifier have not, except for the learning rate. All other hyperparameters remain the same, although they might be overadapted to an OHSM classifier, resulting in a disadvantage for our hierarchical method.

We attribute the faster training to the simpler per-concept decisions made by the hierarchical classifier. While a classification involves many of these decisions, each node is trained simultaneously, resulting in a speed-up.

The hierarchical classifier makes "better mistakes" on NABirds, but it does not on ImageNet-1k. Making better mistakes (*cf.* Bertinetto et al. 2020) is a property of hierarchical classifiers that very likely depends on a strong visual-semantic correspondence (see section 6.6.1). Since the taxonomy of NABirds is based on biology, it is reasonable to assume that many classifications of taxa are based on visual properties of the individuals. In contrast, the WordNet concept hierarchy (see section 3.3.1.1) underlying ImageNet-1k is developed by linguists who focus on semantic distinctions. Consequently, we expect a marginally higher visual-semantic correspondence on NABirds than ImageNet-1k, which would explain this result.

6.7.1.2. Imprecise Data — Supervised CHILLAX

Without inaccuracy, CHILLAX compares favorably to the two baselines leaves only *and* random leaf*. However, the specific advantage depends on the amount of imprecision and fades in cases where the training data is largely precise.* CHILLAX is designed specifically for handling imprecise data. If the training data is completely precise, it behaves identically to the hierarchical classifier from section 5.2, resulting in the same accuracy as for the knowledge integration use case. When imprecise data is present, the advantage over both baselines shows that it is leveraged correctly by CHILLAX.

Even with strong inaccuracy (10 %), CHILLAX still compares favorably to the two baselines leaves only *and* random leaf*. The overall advantage is reduced further compared to* 1 % *inaccuracy, but is still substantial for Poisson-distributed imprecision as well as according to Deng et al. When imprecision is distributed geometrically, CHILLAX only performs best for* $q = 0.95$. The geometric distribution concentrates examples that are labeled as the root of the concept hierarchy, which would be expected in certain web crawling scenarios (see section 5.1.1.1). However, such examples cannot be used by CHILLAX since the loss function is always zero for them, regardless of the prediction (see section 5.3.2.1). This motivates our self-supervised version proposed in section 5.3.3.

The efficiency implied through certain accuracies CHILLAX reaches by learning from training data with a specific IC is similar for imprecision with Poisson and Geometric distribution. The model by Deng et al. implies a comparatively worse efficiency. We discuss the peculiarities of the protocol by Deng et al. in section 6.7.1.3, to which we attribute these differences. Furthermore, the growth of accuracy as IC increases in fig. 6.1 highlights the possible efficiency benefit of imprecise data. As long as its cost is proportional to the information content, a larger increase in accuracy can be bought cheaper at overall lower levels of precision.

CHILLAX always performs better than HEX on ImageNet-1k in terms of top-5 accuracy, following the protocol by Deng et al. However, with respect to top-1 accuracy, HEX produces favorable results for very high imprecision, i.e., $p \geq 0.95$. HEX and CHILLAX are both probabilistic models, however they propose different assumptions. CHILLAX focuses on subsumption informed by a hyponymy relation. HEX also considers subsumption, but adds an explicit exclusion relation, which is non-trivial (see section 5.2.2) for DAG hierarchies such as WordNet (see section 3.3.1.1).

Although there are cases where CHILLAX is not more accurate than HEX, it always makes better mistakes as evidenced by its top-5 accuracy. This property could be due to a stronger semantic influence on the classification process (Bertinetto et al. 2020).

6.7.1.3. Imprecise Data — Self-Supervised CHILLAX

In terms of hierarchical F1, the hierarchy-informed and confidence-based self-supervised CHILLAX variants are able to recover more than half of the information lost to imprecision by extrapolating using predictions. The Deng et al. experimental protocol again produces qualitatively different results to our own noise models. Since the imprecision in the protocol of Deng et al. is limited to the direct hypernyms of precise concepts, recovering the information is in most cases a binary classification. Stronger imprecision leaves more room for actual contributions of the knowledge in the hierarchy and is also more realistic as determined in section 6.6.3.

The hierarchy-informed and confidence-based self-supervised CHILLAX variants perform in different manners. Fixed threshold *achieves the highest accuracies overall, but requires an infeasible amount of fine-tuning.* d^* steps *and* IC range *are only beneficial some cases.* Leaf node *improves the accuracy over CHILLAX in all cases and is a simple heuristic without hyperparameters*. While *fixed threshold* performs best in our experiments, it is impractical to reliably determine the correct threshold while preventing overfitting to a certain validation set (see section 2.2.2.2). In contrast, d^* *steps* and *IC range* have hyperparameters that depend on the concept hierarchy and the actual imprecision in the training data, both of which can be observed. They are also more interpretable. Simply using a *leaf node* is most similar to a typical self-supervised setup such as Wang et al. (2017b), where pseudo-labels are given the highest confidence instead of a more even distribution.

The adaptive self-supervised CHILLAX variant obtains consistent improvements over a wider range of its hyperparameter than fixed threshold*. However, on average,* leaf node *performs better even without any parameterization.* While it is clear that *leaf node* will produce a high fraction of inaccurate examples (see section 6.4.3.1), we also learn in section 6.4.2.2 that CHILLAX presents a reasonable robustness against this type of label noise. Furthermore, we do not observe any catastrophic failures due to feedback loops in our experiments. However, *leaf node* has no mechanism to prevent them, whereas *adaptive threshold* does.

The adaptive threshold *method can successfully control the IC gain, keeping it close to its hyperparameter $\delta\mathfrak{I}^*$.* IC range *exhibits a clear bias towards lower realized IC gains.* The bias of *IC range* can be explained by its asymmetric enforcement of the bounds. While it can prevent any violations of the upper bound, it cannot guarantee the lower bound because a concept might not even exist inside the range. There is no compensation for this bias in *IC range* as there is in *adaptive threshold*, which does not show such a bias in most cases.

If it does, realized IC gain is lower than the target because both methods are fundamentally limited by the actual amount of imprecision present in the training data, which has to be considered when setting hyperparameters.

6.7.2. Webly Supervision

Flickr data: Combining the moth dataset with data downloaded from Flickr increases accuracy from 92.62 % *to* 96.66 % *for CHILLAX, which outperforms the baseline in both cases. However, learning from only the precisely labeled part results in higher accuracies for both methods, and a slight advantage for the baseline.* The OHSM has a slight advantage when there is only precise data, in line with our discussion in section 6.7.1.2. However, the fact that removing data increases accuracy is unique to this dataset. A small sampling of the data shows that search results at intermediate levels in the taxonomy strongly vary in quality, which we address with our filtering setup in section 6.5.1.3.

Using CHILLAX to filter the downloaded images for outliers improves the performance of the baseline, but has a marginal negative effect on the accuracy of CHILLAX. Overall, the Flickr images can be used together with CHILLAX as a filter to increase the accuracy from 92.07 % *to* 97.76 %*.* The fraction of imprecise labels in the filtered data (13.22 %) is considerably lower than in the original download (23.00 %). While the filter is working as intended, and would not be possible without the hierarchical classifier, it does leave CHILLAX too few imprecise examples to work with.

6.7.3. Pitfalls and Dangers

In section 6.6, we present several possible failure cases of hierarchical classification and learning from imprecise data, which we discuss in the following.

6.7.3.1. Visual-Semantic Correspondence

Visual and semantic similarity on CIFAR-100 are (rank-)correlated with a coefficient of $R = 0.23$*, when WordNet is used as a concept hierarchy. The correlation between the semantic baseline, which considers all distinct concepts equally dissimilar, and visual similarity is only* $R = 0.17$*.* Only the difference between these correlations can be leveraged by integrating domain knowledge. While this finding is somewhat sobering, it could be a consequence of the semantically reductive nature of labeling for image classification (*cf.* Brust and Denzler 2019b). Consequently, semantically richer tasks such as image captioning or visual question answering (see section 3.3) possibly benefit more from semantic knowledge integration.

Furthermore, there are machine learning tasks outside of computer vision where the domain gap to semantics is naturally smaller, for example NLP (see section 4.2.5).

6.7.3.2. Harmful Hierarchies

Introducing faulty elements to a concept hierarchy has an adverse effect on accuracy. The effect scales with visual-semantic correspondence. On NABirds, less than three percent points are lost, while on D-CHIVES L, accuracy decreases from 88.98 % *to* 40.96 %*.* This result suggests that, on NABirds, the hierarchical classifier makes little use of visual features relating to imprecise concepts, possibly because not every imprecise concept is represented by a single, independent visual feature from the neural network. On D-CHIVES L, we know constructively that such features exist. The classifier using these features extensively would explain the substantial penalty from introducing faults into the hierarchy. A further possible contributor is the larger number of concepts in D-CHIVES L (4927) compared to NABirds (1010).

We further observe that on D-CHIVES L, the hierarchical classifier and the OHSM baseline reach parity at around $p_{replace} = 0.2$*. However, on NABirds, the OHSM baseline is always superior, even if the hierarchy is not modified at all.* From this mismatch, it could be concluded that the combination of NABirds and its taxonomy is already faulty. Still, we expect that this is largely due to a lower visual-semantic correspondence, at least lower than the perfect D-CHIVES L, and less because of possible mistakes in the taxonomy as the comparison to D-CHIVES L would suggest. Although, while D-CHIVES allows us to isolate the effects of the latter, we cannot distinguish both causes on a pre-existing dataset, *i.e.*, NABirds.

For NABirds, the effect of the level-wise parent replacement procedure on accuracy is more pronounced at $0.0 \leq p_{replace} \leq 0.6$ *than it is for the higher settings. Conversely, on D-CHIVES L, the effect is almost linear.* There are two possible contributors to this effect. First, if the classifier relies less on visual features relating to imprecise concepts on NABirds, as speculated above, the replacements at $p_{\text{replace}} = 0.6$ might already invalidate almost all imprecise concepts whose features are actually utilized. Second, because visual features relating to imprecise concepts are probably not as simple and independently present as in D-CHIVES L, there could be a certain interdependence between features such that a single replacement operation affects predictions w.r.t. to unrelated concepts as well.

6.7.3.3. Imprecision in Real-World Data

Through the lens of the WordNet hierarchy, we observe both the Poisson and geometric distributions over precision in the Flickr metadata. The distributions appear "shifted" towards higher depths. We postulate that this offset in the distributions is due to concepts in WordNet that are so general that they don't exist as real life objects, *e.g.*, `entity.n.01`$^{\text{W}}$ or `abstraction.n.01`$^{\text{W}}$. Still, the empirical distributions in fig. 6.7 don't match the models exactly because WordNet depth is not an ideal representation of precision. In section 5.1.1.2, we discuss an alternative measurement.

The distribution obtained from the images' taglist could appear as a mixture of title and description depths because tags are typically important keywords taken from the other metadata. And while we expect a textual description to be more precise than a short title, it appears that it is not, if it is present at all. In practice, the description is more of an addition to the title giving further context, hence the occurrence of synsets such as `series.n.06`$^{\text{W}}$.

Image title precision is approximately Poisson distributed, which could be due to users attempting to offer a descriptive and artistic caption (*cf.* Brust, Barz, and Denzler 2021b). In contrast, the geometric distribution over the tags' precision could be explained by users optimizing for search engines and using purposely generic words.

6.7.4. Omissions and Limitations

This section briefly lays out and discusses aspects in which this thesis might be considered limited. We offer arguments for our selection of datasets and methods as well as the omission of runtime measurements.

6.7.4.1. Datasets

The central "benchmarking" experiments in this thesis (in section 6.4) focus on the NABirds dataset (see section 6.2.1.2). These experiments involve a number of repetitions to control for random initialization. Furthermore, specifically the experiments in section 6.4.3 have such numerous combinations of interventions and methods that running them on more than one dataset is not feasible computationally.

With this limitation in mind, we select the NABirds dataset for a number of reasons. First, the taxonomy is aligned to scientific consensus in biology and thus unlikely to contain incorrect elements (see section 6.6.2). Second, the benchmark task is a good representation of our recurring example application in biodiversity research, which stands to benefit from any performance improvement that can be obtained without additional high-quality training data.

Scale problems also motivate us to use the CIFAR-100 dataset (see section 6.2.1.1) for the experiment in section 6.6.1. The evaluation requires comparing each image in the dataset with each other with five different methods. For CIFAR-100, this results in 249 975 000 total comparisons which is only possible because of the relatively small-sized images.

Where possible and appropriate, we offer a diverse selection of benchmark datasets, *e.g.*, in sections 6.3 and 6.4.2.4. Furthermore, we construct a synthetic dataset (see section 6.6.2.2) and produce a new dataset from web crawling in section 6.5.

6.7.4.2. Baselines and other methods

Learning from imprecise data (see section 5.1) is comparatively new task, initially published by us in 2021 in Brust, Barz, and Denzler (2021b). The idea of semantic imprecision in general is not new and there are several methods that apply to certain partial aspects of imprecise data (see section 4.2.1). However, there are no methods that solve our specific combination of imprecise training data with precise predictions, other than HEX (Deng et al. 2014), which is limited by its mutual exclusivity assumptions and complex inference procedure. We compare our method to HEX in section 6.4.2.4 using the published results. The experiments cannot feasibly be replicated, as there is no first-party source code available, and the third party implementations are incomplete and obtain substantially worse results[11].

For the simple knowledge integration use case, there are again methods that tackle the problem partially, which we discuss in section 4.1. However, they also suffer from limitations which prevent any application in our intended setting. For example, embedding and metric learning-based methods (see sections 4.1.2 and 4.1.3) that integrate a distance-like semantic measure (see section 6.1.2.2) struggle with hierarchies that are DAGs. For these graphs, shortest paths are not necessarily unique. Furthermore, these methods do not process imprecise concepts at all, except for volume-based methods such as Ganea, Becigneul, and Hofmann (2018) and Dhall et al. (2020).

Since knowledge integration is not the main goal of this thesis, but a step towards enabling learning from imprecise data, we dedicate our resources towards the latter.

6.7.4.3. Runtime Measurements

Since all results in this chapter except for sections 6.5 and 6.6.2 are published over the span of three years, the results are determined using different generations of hardware. Repeating all experiments performed over years on identical hardware for the purpose of determining runtime is not practical, especially considering the limited insight that can be gained from such an investigation.

Our hierarchical classifier (see section 5.2) as well as CHILLAX (see section 5.3.2) both represent only small additions compared to the computational costs of the underlying neural networks (see section 6.2.2.1). In Brust and Denzler (2019a), we conclude that the computational overhead of our hierarchical classifier compared to an OHSM baseline is insignificant. We describe two computational scales in section 6.3.1, which also contains rough estimations of runtimes. These estimations also apply to section 6.4.

---

[11] `https://github.com/kylemin/HEX-graph` is the most complete implementation as of July 20th, 2021. It reaches 54.10 % top-5 accuracy on ImageNet-1k, compared to the original at 68.50 % and CHILLAX at 68.55 %. `https://github.com/ronghanghu/hex_graph` is no longer available.

7. Experimental Outlook Towards Lifelong Learning

Data scarcity is the main reason why learning from imprecise data is important. Not fully utilizing training data can be an unaffordable luxury. To explore the idea of increasing sample efficiency further, we take a step back from the learning process and consider the actual labeling process. Active learning assumes that not every unlabeled element of data is equally worthy of being labeled (*cf.* Settles 2009).

Combined with an incremental learning method that can gradually improve a model over time, active learning can be integrated into a feedback loop. Such a system continuously explores a set of unlabeled data and presents valuable examples to a human labeler. After labeling, the model can learn from the new examples and continue the exploration in light of its new knowledge. Repeating this process continuously is called *lifelong learning*.

7.1. A Brief Introduction to Lifelong Learning

This section serves as a brief overview of the theoretical foundations of lifelong learning. For contrast, consider a typical application of machine learning, which involves several steps. First, training data has to be acquired, *e.g.*, by taking pictures and annotating them. Afterwards, one needs to decide on an appropriate machine learning method, or set of methods. The next step is model selection (see section 2.2), where the optimal risk-minimizing hypothesis and hyperparameters are determined. Finally, the hypothesis has to be validated using a held-out test set (see section 2.2.2.1), and then deployed for productive use.

This process, also known as "waterfall" learning (see Data Science Process Alliance 2021) is straightforward, but suffers from a number of drawbacks. Mainly, there is no consideration for the lifecycle of the hypothesis after its creation. The same is true for the lifecycle of the respective training data. However, training data can become less representative of the real world, or the environment distribution, over time. The appearance of objects changes, and new concepts are introduced as well. A further disadvantage of waterfall learning concerns the human and computational resources. It requires a considerable amount in the beginning of a project, when training data is acquired and hypotheses are tested, and afterwards, only a very small amount for maintenance and operation. This distribution of resources over time is not ideal for long-running projects such as biodiversity monitoring.

Lifelong learning addresses the aforementioned lifecycle aspects. It considers availability of new data over time and integrates with data streaming processes, *e.g.*, camera traps (see section 7.3.2). The hypothesis is continually adapted to the changing environment. In the following sections, we review the individual components of a lifelong learning systems, including active learning and incremental learning. We then describe the assembly into a complete system, and the construction of the lifelong learning cycle.

7.1.1. Active Learning

Training data is obtained from the environment distribution by sampling $(x, y) \sim \mathcal{D}$ (see section 2.1.1.7). In practice, there are two steps. First, the domain points x are acquired, *e.g.*, by taking pictures or collecting other measurements. Then, they are annotated with labels y by human annotators.

Active learning acknowledges this separation and seeks to leverage it. It hypothesizes that not all x are equally worthy of annotating (*cf.* Settles 2009). Consider a set $X = (x_1, \ldots, x_{m'})$ of m' not yet annotated domain points. We then fix an *annotation budget* of $m << m'$. The goal of active learning methods is to select an optimal in-budget subset $X_{\text{opt}} \subset X$, where $|X_{\text{opt}}| = m$. This subset is optimal in terms of true risk of a hypothesis resulting from annotating X_{opt} and applying an ERM (see section 2.1.1.3) learner. Ideally, the risk is lower than from *passive learning*, *i.e.*, randomly selecting a subset.

However, the active learning problem of optimal subset selection is fundamentally ill-posed. It has no access to the annotations because they have not yet been acquired. However, without annotations, it is not possible to estimate the risk associated with any given subset. Hence, several heuristics are proposed as alternatives. *Uncertainty* is a common choice (*cf.* ibid.). Assume that there exists a hypothesis that has not yet seen X, but is configured correctly for the domain, *i.e.*, with the correct input dimensionality and number of concepts. We can then apply it to an unlabeled example and analyze confidence scores, or predicted probabilities, for each concept (see sections 2.1.2.2 and 5.3.3). A unimodal distribution indicates that the hypothesis is very confident, while a uniform distribution represents maximal uncertainty. Hence, entropy is a reasonable indicator of uncertainty (*cf.* Wang et al. 2017b). A simpler alternative is *1-vs-2*, which is the difference between the confidence scores of the highest-scoring concept and the second highest.

These indicators are denoted *value functions* $v(x, h)$. The optimal subset selection of active learning is performed by computing the value function for all $x \in X$ and then selecting the m most valuable examples. For example, given per-concept probabilities h_c, the value function of 1-vs-2 is defined as (*cf.* ibid.):

$$v(x, h) = 1 - \big(\max_{c_1 \in \mathcal{Y}^P} h_{c_1}(x) - \max_{c_2 \in \mathcal{Y}^P \setminus c_1} h_{c_2}(x) \big) . \tag{7.1}$$

Examples that strongly influence a model's parameters, or its predictions, are also considered valuable, as the changing parameters are associated with "fast" learning progress. This heuristic is discussed further in Freytag, Rodner, and Denzler (2014) and Käding et al. (2016a).

Novelty detection is a related task which specifically considers examples containing previously unseen concepts. While it can be performed as a "side effect" of active learning, there is research into isolated novelty detection (*e.g.*, Schölkopf et al. 1999; Bodesheim et al. 2013).

7.1.2. Incremental Learning

In a system where new training data becomes available over time, a hypothesis has to be updated frequently. Because predictions are used in active learning to select valuable examples for annotation, updates should not be delayed. Otherwise, redundant examples or even repeated selections can occur. However, training high-performance models such as CNNs is expensive and time-consuming (see section 2.3.3.2). Furthermore, the training time increases with the amount of training data, which makes constant re-training from an initial state infeasible at some point.

Incremental learning methods tackle the problem of adding new training data to an existing model. One particular challenge is *catastrophic forgetting* (*cf.* Kirkpatrick et al. 2017): while the risk w.r.t. new training data becomes better, the risk of the hypothesis over the previous training data increases.

In this work, we use the *deep fine-tuning* method described in Käding et al. (2016b) for deep neural networks. It is an approach based on stochastic gradient descent (see section 2.1.3.1), where the initial parameters are copied from the current hypothesis, as opposed to a random initialization. To update the hypothesis, a fixed amount of optimization iterations is performed. Crucially, each minibatch is composed of both new and already known training data to combat catastrophic forgetting. The ratio of old to new, λ, can be optimized as a hyperparameter, but is usually set to 0.50.

Concept hierarchies are related to incremental learning in two distinct ways. On the one hand, structural hierarchical classifiers (see section 4.1.1) can implement incremental learning (*cf.* Roy, Panda, and Roy 2020). On the other hand, concept hierarchies can define an ideal order (a "curriculum") in which examples should be learned to minimize risk (*cf.* Goyal and Ghosh 2020).

7.1.3. The Lifelong Learning Cycle

This section lays out the lifelong learning cycle as proposed in Käding et al. (2016a). It is presented as an alternative paradigm to waterfall learning (see the introduction to section 7.1).

Initialization The cycle starts with certain initial conditions. An initial set of training data $S^{(0)}$ has to be sampled randomly and annotated manually. This data is needed because the active and incremental learning components require a functioning hypothesis $h^{(0)}$ that has lower true risk than a random hypothesis. However, there exists preliminary research around removing this requirement for initial training data in Penzel (2018).

Cycle After initialization, the cycle is entered. The following steps are repeated indefinitely, or *lifelong*.

> Iteration k starts with acquiring unlabeled examples. All unlabeled examples available in this iteration are denoted $X^{(k)}$. We have access to the hypothesis $h^{(k-1)}$ generated during the last iteration.

1. **Exploration.** Compute the value of each unlabeled example using an active learning value function $v(x, h)$. Determine $X_{\text{opt}}^{(k)}$ by selecting the m most valuable examples.
2. **Interaction.** Acquire labels for $X_{\text{opt}}^{(k)}$ from a human annotator to construct training data $S^{(k)}$.
3. **Adaptation.** Update the hypothesis $h^{(k-1)}$ with $S^{(k)}$, but without forgetting $S^{(k-1)}, \ldots, S^{(0)}$ using incremental learning. This results in the improved $h^{(k)}$.

The lifelong learning cycle has constant human interaction and compute resource requirements over time, which is suitable for long-running projects. If the availability of resources changes, m can be changed at any point to adapt to the changes. Note that we do not consider the memory complexity explicitly and assume an infinitely increasing set of unlabeled examples $X^{(k)}$. In a practical application, the process should remove unlabeled examples after they have not been selected for some time to guarantee infinite execution with limited resources, at least in theory. A similar process could be applied to training data, where examples could be removed when they become outdated. All experiments and applications in the following sections are constructed following the lifelong learning cycle described here.

7.2. Active and Incremental Learning for Object Detection

Partial results of the work presented in this section are published in Brust, Käding, and Denzler (2017, 2019, 2020).

Object detection is a particularly interesting task from the perspective of lifelong learning in general and active learning specifically. The complexity of the labels and the higher cost present unique challenges and opportunities. While active learning methods for classification are not directly applicable to object detection, the potential for cost savings is much greater. There are also opportunities for improvements of the human-machine interaction aspect (see section 7.3).

In this section, we use YOLO (You Only Look Once, *cf.* Redmon et al. 2016) to apply our active learning methods, which are specific to object detection. While most are applicable to any detector, we also present bespoke approaches for YOLO. All methods are validated empirically in a lifelong learning setting on the PASCAL VOC (Visual Object Classes, *cf.* Everingham et al. 2010, 2015) dataset.

7.2.1. Active Learning for Object Detection

Most active learning methods assign a value to each example in a set of unlabeled data (*cf.* Settles 2009). For image classification tasks (see section 2.3.2), an example is typically defined as a single image. While object detection would allow for instances of objects to be considered as examples, we adhere to the former definition because instances are not known before labeling. It should be noted that predictions from a pre-trained detector could be used to provide these instances. Another reason for using images as examples is the increased efficiency of annotating entire images at once.

In the following, we propose two types of active learning value functions, one for object detection in general, and one specific to the YOLO detector.

7.2.1.1. Aggregation of Individual Detections

One active learning strategy in classification is to analyze the predicted probability distribution (see section 2.1.2.2) of a hypothesis given the image in question (*cf*. Wang et al. 2017b). This strategy fits well with lifelong learning where such a hypothesis is updated continually. We can interpret the prediction of a detector given an image as a collection of individual predictions, each classifying a detected instance. For the purpose of active learning, we ignore the localization aspect. There exists preliminary research on active learning of localization in terms of a regression task in Käding et al. (2018).

For each detected instance, we can calculate the 1-vs-2 value from the predicted class probabilities using eq. (7.1). We then aggregate the values into one for the whole image by taking the sum, the average or the maximum.

7.2.1.2. YOLO-Specific Metrics

YOLO, presented in Redmon et al. (2016), is a significant improvement upon previous object detectors (*e.g.*, Girshick 2015). Instead of treating localization and classification as separate, subsequent tasks, they are performed simultaneously. This mode of operation is called *single-stage* or *single-shot* detection and is also used by SSD (Single-Shot Multibox Detector, *cf*. Liu et al. 2016).

A fundamental problem for deep learning-based single-stage detection is the encoding of a variable amount of instances in a fixed size CNN. YOLO solves the problem by dividing the image into a grid of fixed size, *e.g.*, 5×5 or 7×7 (*cf*. Redmon et al. 2016). Each grid cell is classified unconditionally, *i.e.*, even if there is no object present. In addition, YOLO predicts an “objectness” score and bounding box dimensions twice per grid cell. Bounding boxes are always centered in the respective grid cell, and there is no prediction of locations, only dimensions. For a 7×7 grid, there are 49 classification and 98 potential bounding boxes. A threshold on the objectness is used to filter the predicted bounding boxes, resulting in a variable number of predicted instances. Non-maximum suppression is applied to remove multiple detections of the same object (*cf*. ibid.).

In general, we expect a high maximum class score and a high objectness to be predicted at the same time. Our first YOLO-specific metric, “Detection-Classification Difference”, selects cases where the scores disagree strongly, in terms of absolute difference between maximum class score and objectness. Disagreement points to either missed localizations or new classes, both of which are interesting in terms of active learning. We also propose “Weighted Cell Sum”, where we compute a 1-vs-2 value for each grid cell’s classification. The values are weighted according to their highest objectness score and the added together. Assuming perfect localizations, this is equal to taking the sum as proposed in section 7.2.1.1, ignoring the non-maximum suppression.

7.2.2. Incremental Learning of YOLO

The incremental learning method presented in Käding et al. (2016b) is directly applicable to YOLO. However, special care needs to be taken when new classes are encountered. Weights in the last layer of the CNN have to be rearranged depending on the memory layout of the YOLO implementation. Assume the model is configured with a 7×7 grid, two bounding box predictions per grid cell and 20 possible classes. The number of weights per grid cell is then 20 (class scores) + 2 (bounding box predictions) × $\left[\right.$ 4 (dimensions) + 1 (objectness) $\left.\right]$ = 30. The last layer has to be configured with 1470 neurons.

Adding a new class means extending this layer by 49 more weights. We initialize the new weights with zero. Because YOLO does not use a softmax activation function, a zero weight actually means the new class is never predicted unless the weight changes.

7.2.3. Experiments

We validate all methods described in sections 7.2.1 and 7.2.2 empirically. This quantitative evaluation implements a full lifelong learning cycle (see section 7.1.3) to give the methods an appropriate context. We further provide a qualitative analysis of the active learning value functions on individual samples of the dataset.

7.2.3.1. Setup

The experiments are performed using the PASCAL VOC dataset. The dataset is presented alongside a challenge in Everingham et al. (2010, 2015). We use mean average precision (mAP) as described *ibid.* to evaluate the object detection performance.

To simulate a lifelong learning application, we split the dataset in two. This affects both training and validation data in the same manner. One part (A) is used to train and validate the initial model. The other part (B) is then explored during the lifelong learning cycle. We split the dataset by classes to force encounters of unseen concepts. However, the assignment of classes to part A or B can strongly influence the results. Hence, we employ two ways of splitting the data. The first split assigns `bird`, `cow` and `sheep` to part B and the rest to part A, while the second split uses `tvmonitor`, `cat` and `boat` as part B and the remainder for part A.

We implement the "YOLO-Small" architecture exactly as described in Redmon et al. (2016). Following their implementation, the weights of all but the last layer are copied from an "Extraction" model[1] pre-trained on the ImageNet-1k dataset (see sections 6.2.1.1 and 6.2.2.2).

The initial model is trained for 24 000 iterations using the Adam optimization algorithm presented in Kingma and Ba (2014) (see section 2.1.3.3). We use a learning rate of 1×10^{-4} for the first half of this training and 1×10^{-5} subsequently. All other hyperparameters are taken from Redmon et al. (2016), including data augmentation. Each method — sum, maximum, average and the two YOLO-specific methods — is evaluated in five runs for each of the two ways of splitting the data. A random selection baseline is added for comparison.

---

[1] Provided by the authors at `http://pjreddie.com/media/files/extraction.conv.weights`

7.2.3.2. Implementation of Learning Cycle

After initial training, the model is ready for executing the lifelong learning cycle. Before, the roughly 600 training examples of part B are randomly assigned to batches of size 10. These assignments are not changed during the run. Instead of individual examples, a single batch is selected for each iteration of the cycle. The active learning value of a batch is determined by the sum of the values of the examples. We then "annotate" the batch with the correct labels from the dataset. Finally, the model is updated using the fine-tuning method presented in Käding et al. (2016b) for 100 iterations. Hyperparameters match those given in section 7.2.3.1. After removing the selected batch from part B, the cycle is repeated until exhaustion, *i.e.*, after 60 batches.

7.2.3.3. Results

Catastrophic forgetting (*cf.* Kirkpatrick et al. 2017) is a concern in any application of incremental learning. Hence, we observe not only the performance on the validation set of part B (the "new" data) but also on part A, which is used to train the initial model. In the worst single run, the mAP decreases slightly, from 36.70 % down to 32.10 %, but is otherwise not strongly affected.

Table 7.1 shows the performance on part B, the new data, as it is explored. We mainly focus on the first half of the experiment, *i.e.*, the first 30 batches. This scenario is more relevant for practical applications, as it reflects situations of high sample diversity in the unlabeled data pool. Continuous streaming of new data, *e.g.*, from camera traps, keeps the diversity up. W first investigate the performance after 50 samples of the part B training set. Both maximum and average aggregations outperform the random baseline in terms of mAP, while the simple sum and "Detection-Classification Difference" perform even worse. The YOLO-specific "Weighted Cell Sum" has the highest mAP.

Over the next 200 samples we observe continuously improving performance, indicating that the incremental learning works as expected. We also observe that the random baseline is outperformed by most methods more and more as training continues. A notable exception is "Detection-Classification Difference", which performs worst in all comparisons.

For a complete picture, we also analyze the performance after 600 samples, where there is no more choice for the active learning methods. The final mAP is almost equal between methods because they can only affect the order in which examples are learned, but are forced to learn all eventually. Instead, we can observe the area under the learning curve (AULC), where the learning curve is defined as mAP sampled every 5 batches. Again, all methods outperform the random baseline, except for "Detection-Classification Difference". Maximum aggregation is best by a small margin, followed by both sum-based methods. We discuss our findings in section 7.2.4.

7.2.3.4. Sample Valuation of Initial Model

After training an initial model on part A of the dataset as described in section 7.2.3.1, we evaluate the active learning value of each example in part B using the proposed

Table 7.1.: mAP (%) and AULC on validation set of part B of split PASCAL VOC 2012 dataset. *DCD* and *WCS* refer to *Detection-Classification Difference* and *Weighed Cell Sum*, respectively. Observed samples *excluding* initial training data from part A. Table compiled of data from Brust, Käding, and Denzler (2019, 2020).

Observed samples	50	100	150	200	250	All (600)
Baseline						
Random	8.7/4.3	12.4/14.9	15.5/28.8	18.7/45.9	21.9/66.2	32.4/264.0
YOLO-spec.						
DCD	8.5/4.3	12.1/14.6	15.5/28.4	18.7/45.5	21.0/65.3	**33.3**/255.3
WCS	**9.6**/**4.8**	12.9/**16.1**	16.6/30.8	**20.5**/49.4	21.9/70.6	32.2/268.1
Aggregated						
Max	9.2/4.6	12.9/15.7	15.7/30.0	19.8/47.8	22.6/69.0	32.0/**269.3**
Avg	9.0/4.5	12.4/15.2	15.8/29.2	19.3/46.8	**22.7**/67.8	33.3/266.4
Sum	8.5/4.2	**14.3**/15.6	**17.3**/**31.4**	19.8/**49.9**	22.7/**71.2**	32.4/268.2

aggregation methods. Figure 7.1 shows the most valuable and least valuable examples according to each method.

The least valuable examples are the same for each method because they do not contain any detections at all, resulting in a zero score. The sum aggregation prefers images containing many instances. In contrast, the most valuable examples according to maximum and average values mostly have one instance. Because both aggregations are equal in this case, they share many of the top images.

7.2.4. Summary and Discussion

Assuming that labeling images with more objects in them is more efficient in terms of instances per time period, we recommend using one of the sum-based aggregation methods. Because they prefer images with many instances, the hypothesis is changed more significantly in each iteration of the learning cycle. Still, we do not evaluate the exact labeling cost for each method, which should be considered in practice.

During most of the time in the experiment, maximum and average aggregation perform very similarly. This is likely due to the prevalence of single-instance examples in PASCAL VOC 2012. We can not recommend "Detection-Classification Difference", which also exhibited an AP (see section 6.1.1.1) of less than 0.50 in a related *novelty detection* experiment. In other words, the inverse of this method is probably a better choice.

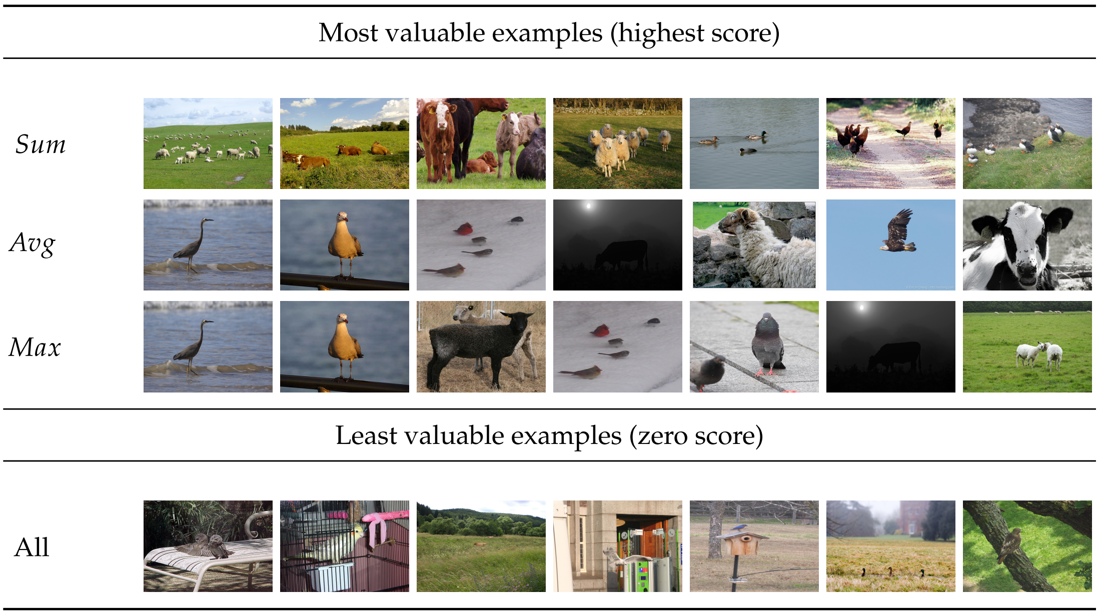

Most valuable examples (highest score)
Sum
Avg
Max
Least valuable examples (zero score)
All

Figure 7.1.: Most and least valuable examples of validation set of part B of split PASCAL VOC 2012 dataset. Figure taken from Brust, Käding, and Denzler (2020).

7.3. Weakly Supervised Lifelong Learning for Object Detection

Partial results of the work presented in this section are published in Brust, Käding, and Denzler (2020).

Active learning is typically employed in situations where labels are prohibitively expensive, thus mandating a small, but informed, selection. However, active learning only considers whether an image should be annotated or not. The quality of the annotations and the human-machine interaction to acquire them is a separate issue. Object detection is a good task to explore these areas given the high complexity of the labels and their large number of degrees of freedom.

7.3.1. Interaction Model

Let us first consider two qualitative extremes of annotations for object detection. On the one hand, images labeled with perfectly localized bounding boxes are an expensive, but important ingredient for an accurate model. On the other hand, it is also possible to learn object detectors from images that are only annotated with class labels, without any location information. However, this has a strong negative impact on performance (*cf.* Song et al. 2014).

In Papadopoulos et al. (2016), the authors propose that neither of these extremes is optimal. Instead, they offer a trade-off: use a pre-trained detector to generate proposals, and ask a human annotator only for verification of these proposals. While a slightly higher number of annotations is needed to reach comparable performance on a validation set, the overall speedup in that case is still in the range of 6 to 9. An important ingredient is a tight feedback loop, where the model is updated continuously. Hence, we consider this annotation process a very good fit for lifelong learning applications, which provide the feedback loop through iterations of the cycle, and also specify a human-in-the-loop (see section 7.1.3).

Trading off label quality and quantity in this way is an example of weakly supervised learning, which is described in more detail in sections 2.1.4.3 and 4.2.4.2.

7.3.2. Camera Traps

Monitoring the abundance and regional occupancy of animals is an important task in light of concerning global developments (see chapter 1). Camera traps are an increasingly popular alternative to invasive and expensive monitoring methods like capture-mark-recapture (Amstrup, McDonald, and Manly 2010). However, they can easily produce more data than can be processed manually. Accurate models require a considerable annotation effort, some of which can be performed by citizen scientist, *e.g.*, as detailed in Swanson et al. (2015). The specific challenges of camera trap data are an increasingly popular research subject. Further works tackling these are Giraldo-Zuluaga et al. (2017, 2019), Gomez Villa et al. (2016), Gomez Villa, Salazar, and Vargas (2017), and Norouzzadeh et al. (2017).

7.3.3. Experiments

In the following, we detail a first feasibility study of propose-and-confirm annotations (as shown in Papadopoulos et al. 2016) integrated into a lifelong learning system for object detection (see section 7.2). The experimental setup is identical to the one in section 7.2.3.1. We use the (non-YOLO-specific) sum aggregation to select batches.

7.3.3.1. Dataset

The dataset used for this experiment is unpublished. It is part of an iDiv[2] study on the impact of large herbivorous mammals on forest development. 65 camera traps are deployed in $\sim$ 16 km^2 and collected data over 3 to 4 months in 2015 and 2016. The traps are located in Peneda-Gerês National Park in northern Portugal. The dataset consists of 1 500 000 unlabeled images of 15 mammalian species.

There are numerous challenges (see fig. 7.2) such as occlusions, intentionally camouflaged animals, motion blur, large herds and strong lighting differences caused by the day-night cycle. At night, the images are illuminated by infrared, which results in a complete loss of color information. The cameras can also be triggered spuriously, without an animal present, by lighting changes, plant movements due to winds and also researchers traversing the area.

7.3.3.2. Evaluation

There are no bounding box annotations available for this dataset. However, 5000 images are supplied with class labels for the whole image. We use these images to

---

[2] German Centre for Integrative Biodiversity Research

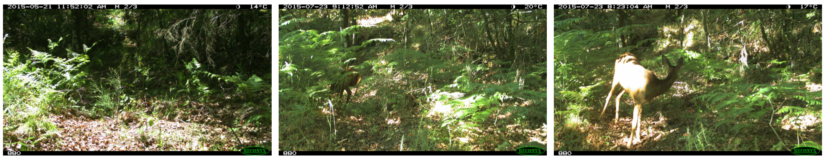


Figure 7.2.: Camera trap images of varying quality from Peneda-Gerês National Park. Figure taken from Brust, Käding, and Denzler (2020).

validate our model, as there is no viable alternative. The instances detected by the model are transformed to multi-label classifications. Localization is hence not considered in this evaluation. However, it should be noted that the intended application is the counting of animals to estimate species abundance, where localization is not necessary.

We further use 5000 unlabeled images for training, and select batches of 32 images for annotation each iteration of the lifelong learning cycle.

7.3.3.3. Results

As described in section 7.2.3.1, the initial model is trained on the PASCAL VOC 2012 dataset (Everingham et al. 2010, 2015). There is some class overlap with our camera trap dataset. Hence, even the initial model can make some accurate predictions. After mapping each class in PASCAL VOC 2012 to its counterpart in the camera trap data, the accuracy is 66.50 %.

We then run the lifelong learning cycle for 16 batches, *i.e.*, 512 images, updating the model after each batch is labeled by an expert and selecting new images using the sum aggregation method. This increases the accuracy to 78.70 %.

However, it is important to note that only 37.80 of the examples in the validation set contain any objects. This is representative of the actual camera trap application because of spurious triggers of the camera (see section 7.3.3.1). Still, we should analyze the non-empty portion of the validation set separately. There, the initial model reaches an accuracy of 25.40 %. After only one batch, *i.e.*, 32 annotated examples, the accuracy is already increased to 42.60 %. It finally reaches 58.50 % after 512 images are annotated by the expert.

The scope of this experiment is limited because of the low availability of expert annotations. However, we consider it a first validation of the combination of weakly supervised learning and lifelong learning.

7.3.4. Carpe Diem Annotation Tool

Partial results of the work presented in this section are published in Brust, Barz, and Denzler (2021a).

As a practical application of the propose-and-confirm interaction model, combined with lifelong learning, we develop a graphical user interface. It is intended for biodiversity researchers, and is thus easy and intuitive to use (see fig. A.2 in the appendix). We name the tool “Carpe Diem”, in reference to the original meaning of YOLO (Redmon et al. 2016). Carpe Diem implements all steps of the lifelong learning cycle and focuses on human-machine interaction. It allows the user to experience a continually improving model for regular analyses of camera data.

We use the CN24 deep learning framework presented in Brust et al. (2015a), which implements YOLO (Redmon et al. 2016) in a way that can cope with the addition of new classes. This process is described in detail in section 7.2.2. The central object in Carpe Diem is a project. A project contains the current model and both unlabeled and labeled data pools.

The first step when starting a new project is the import of unlabeled data. Batches are automatically assigned randomly, but the batch size is controllable by the user so that they can choose the length of an annotation session. When the user is ready to annotate, an active learning method of their choice is executed on all unlabeled data, selecting the best batch for annotation. The interaction can be done with keyboard shortcuts or the mouse. If the proposal is either completely correct or a spurious localization, then a single click is enough for annotation. The class label can be corrected by entering the first few letters of the correct label, aided by auto-completion.

A model update is performed after annotating each batch. On an NVIDIA GeForce GTX 970 (from 2014), a model update of 32 examples requires 3200 iterations and completes in around one hour. This GPU can compute $\sim$ 4 TFLOP/s, compared to current (2021) performance of GPUs on the order of $\sim$ 150 TFLOP/s (see also section 2.3.3.2).

7.4. Label-Efficient Gorilla Re-Identification

Partial results of the work presented in this section are published in Brust et al. (2017).

The effects of the global biodiversity crisis are readily apparent when analyzing threatened species (see chapter 1). Conservation efforts are essential to countering this development. The effects need to be monitored closely to validate individual efforts and to guide further strategies. However, biodiversity monitoring is both expensive and challenging on a technical level (see, *e.g.*, section 7.3.2 and Kühl 2008).

Great apes are critically endangered (*cf.* Vié, Hilton-Taylor, and Stuart 2009). They are also unusually difficult to monitor and hard to find in the first place (*cf.* Kühl 2008). Still, monitoring them is rewarding as it aids research not only w.r.t. biodiversity, but also on their behavior and sociodemographics. Fine-grained monitoring on an individual level is required for many of these studies. It is also the subject of the following experiment.

Previously, in section 7.3, we acquire images from camera traps and use them to estimate the abundance of certain species. Automatic camera traps are not always suitable for individual identification. They produce a low fraction of usable images and require prohibitive amounts of processing (*cf.* ibid.). Instead, great apes are monitored by field photography. Researchers observe the population from a viewing platform and take pictures using a long telephoto lens. This process results in more detailed observations and requires less processing. In the following, we describe a system to identify western lowland gorillas (*gorilla gorilla gorilla*) using a combination of face detection and classification based on field photography.

7.4.1. Re-Identification by Face Detection

Gorillas are commonly identified via their facial features such as the shape of their ears, crest and brow-ridge (*cf*. Parnell 2002). Thus, we propose to base our automatic identification system on the individuals' faces as well. We apply a face detector and then classify the extracted faces. This system is an extension of previous work on chimpanzees presented in Freytag et al. (2016).

7.4.1.1. Face Detection

Any subsequent identification depends on the successful extraction of faces from an image. Hence, a reliable face detector is essential. In the supplementary material of Freytag et al. (ibid.), the authors supply a chimpanzee face detector that shows promising results. We base our detector on the same model.

It is a YOLO model (refer to section 7.2 and Redmon et al. (2016) for details). Modifications include a high-resolution input (448×488) and a larger grid size (9×9) to allow for the detection of more, smaller objects. While YOLO could also perform classification, the training data requirement would be too high because of the spatial dependency of YOLO's classifier. We rebuild this model in CN24 (Brust et al. 2015a) to allow for incremental learning.

7.4.1.2. Individual Identification

Quick model updates in the field are essential to a monitoring system that is meant for daily use. Hence, we rely on a combination of CNN features and SVM classification. The deep neural network is only trained once and then used as a constant feature extractor. When new observations are added, the SVM can be trained in less than a second on commodity hardware. This combination is shown to be effective in Freytag et al. (2016).

We extract features from the `pool5` layer of a BVLC AlexNet variant which is described in Jia et al. (2014). This AlexNet is pretrained on ImageNet-1k (see section 6.2.1.1). While more face-specific models are available (*e.g*. Parkhi, Vedaldi, and Zisserman 2015), they tend to perform worse for chimpanzee classification (*cf*. Freytag et al. 2016). All pre-processing steps are equal to Jia et al. (2014) to permit a direct application. A linear SVM performs classification to identify the faces by their extracted features. We use the LibSVM (Chang and Lin 2011) implementation. Scores are calculated for all possible individuals to obtain a ranking by confidence.

7.4.2. Experiments

To validate the effectiveness of our system and its individual components, we perform a quantitative evaluation. We first measure the sample efficiency of adapting the chimpanzee detector from Freytag et al. (2016) to gorillas by fine-tuning (see section 6.2.2.2). Then, we evaluate the accuracy of individual identification on a large dataset, involving the complete system.

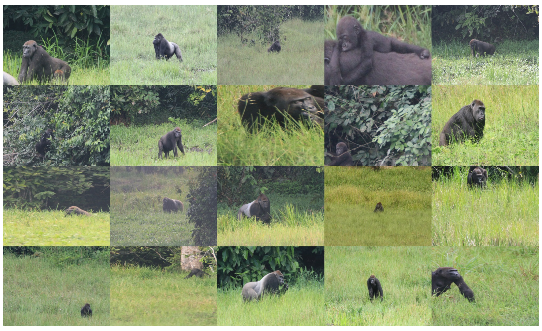


Figure 7.3.: Example photographs of the Mbeli Bai dataset. Figure taken from Brust et al. (2017).

7.4.2.1. Dataset

Our dataset consists of observations of western lowland gorillas (*gorilla gorilla gorilla*) in Mbeli Bai. The Bai is a large forest clearing in Nouabalé-Ndoki National Park in the Republic of Congo, where a population of 129 individuals lives. Observations start in 2012 and end in 2017, resulting in a total of 12 765 photographs (see fig. 7.3).

Each photograph is annotated by two individual observers who determine the identity of the individual in focus. This system guarantees high-quality labels. However, there is only one annotation per image (even though there might be multiple individuals), and there are no bounding boxes.

7.4.2.2. Face Detection Results

To evaluate the detection performance of the YOLO-based chimpanzee detector (Freytag et al. 2016), and to fine-tune the detector for gorilla faces, we commission bounding box annotations for 2500 images. Of these, 500 are held out for validation. We fine-tune the chimpanzee detector separately on 500, 1000, 1500 and 2000 of the remaining gorilla images. Each fine-tuning consists of 3500 iterations of the Adam optimization algorithm (Kingma and Ba 2014) with a learning rate of 1×10^{-5}.

The detailed results on the held-out validation set are presented in the appendix, in table A.8. Initially, the chimpanzee detector by Freytag et al. (2016) reaches an average precision (AP) of 29.50 %. However, after fine-tuning it on only 500 images of gorillas, the AP immediately increases to 86.60 %. Further improvements are small, reaching a final APs of 90.80 % with 2000 annotated image.

7.4.2.3. Individual Identification Results

Using the best performing face detector from the previous section, we can evaluate our system's individual identification accuracy. We use five-fold cross validation (see section 2.2.2.2) to train a linear SVM using the hyperparameters specified in Freytag et al. (ibid.).

There is a concern when performing the validation per image and not per face. In rare cases (4.10 %, see table A.8), the detector produces more than one detection. This is correct, as photographs are not constrained to a single individual. However, there

Table 7.2.: Individual identification performance on Mbeli Bai validation sets. Training images refers to the amount used to train the face detector. Table compiled of data from Brust et al. (2017).

# Training Images	—	500	1000	1500	2000
Highest Score					
Accuracy (%)	20	59	60.3	61	61.7
Precision (%)	48.6	55.5	57.9	58	59.5
Recall (%)	20.3	59.9	61.7	62.3	62.5
Top-5 Accuracy (%)	27.6	77.6	78.9	79.3	79.6
Largest Box					
Accuracy (%)	20	58.8	61.3	62	**62.4**
Precision (%)	48.7	55.5	58.7	59	60.2
Recall (%)	20.3	59.9	62.5	63.2	63.1
Top-5 Accuracy (%)	27.8	77.4	79.5	80.1	80.3

is only one annotated individual per image, which is supposed to be the individual in focus. We propose two heuristics. We either use the bounding box with the highest confidence value, or use the bounding box with the largest area.

In the following, we evaluate both heuristics separately. Table 7.2 shows the results over the validation splits. Note that any errors here can occur because of either misclassification or misdetection, as we have to rely on the detector to correctly extract faces. With the initial chimpanzee detector, both methods achieve an accuracy of 20.00 % and a top-5 accuracy of around 27.70 %. Using the detector trained on only 500 images, selecting the most confidently detected face results in an accuracy of 59.00 % (top-5: 77.60 %). The largest box heuristic with the same detector reaches only 58.80 % (top-5: 77.40 %). Finally, we apply the best performing detector (2000 images). “Highest confidence” and “largest box” deliver accuracies of 61.70 % (top-5: 79.60 %) and 62.40 % (top-5: 80.30 %), respectively.

7.4.3. Discussion

A reliable detector for *gorilla gorilla gorilla* faces can be built with very little annotation effort. Even our smallest labeled batch of 500 images results in comparable performance to the best result. Additional bounding boxes yield only small returns. This detector could be reused for related species, as results from using a chimpanzee detector on gorillas indicate.

In this system, new individuals can be introduced easily and retraining any large models is not required. While field photography typically generates fewer images than automatic camera traps in a given time period, a large enough operation might still overwhelm human annotators. Active learning could be applied directly to reduce load and increase sample efficiency.

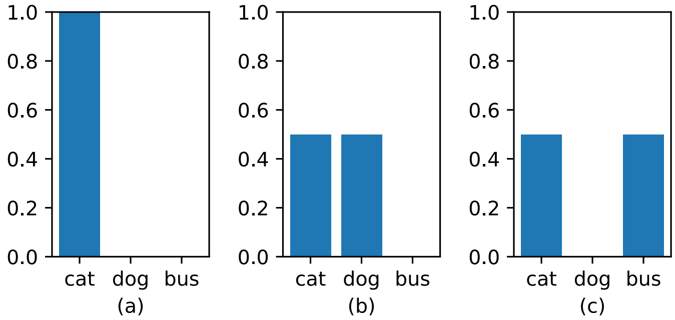


Figure 7.4.: Predicted probabilities of a hypothesis w.r.t. three different domain points.

7.5. Lifelong Learning with Concept Hierarchies

This section briefly lays out a potential application of concept hierarchies in a lifelong learning setting. We already discuss implications of concept hierarchies for incremental learning in section 7.1.2, *e.g.*, constructing a curriculum. Active learning also stands to benefit from integration of semantic knowledge. Birodkar, Mobahi, and Bengio (2019) states that the redundant examples, which active learning seeks to eliminate, have unique semantic properties that differentiate them from valuable examples. However, the authors don't translate their findings into an active learning method and simply claim that their "findings may also be of interest to active learning community." (sic!)

To integrate semantic knowledge into active learning, we propose *hierarchical 1-vs-2*. It is based on the observation that a CNN's confusion probability between two concepts (see section 4.1.5) is correlated to their semantic similarity (see section 6.1.2.2). Hence, it is unlikely that a CNN confuses two concepts that are semantically distant. We assume that this unlikely occurrence indicates an edge case in the data, and thus, a valuable example.

Consider the confidence scores shown in fig. 7.4. The conventional 1-vs-2 method would assign a low value to example (a) because the prediction appears very confident. However, it would also treat examples (b) and (c) equally, even though they are qualitatively different. An image that produces similar scores for `cat` and `dog` is reasonably valuable and annotating it might slightly improve the discrimination between these two concepts. But an image that is equally likely to contain a `bus` or a `cat` must be more "groundbreaking" and result in a fundamental change in the hypothesis.

We define the hierarchical 1-vs-2 value function as:

$$v(x,h) = \mathfrak{d}(c_1,c_2)\left[1 - \left(\max_{c_1 \in \mathcal{Y}^P} h_{c_1}(x) - \max_{c_2 \in \mathcal{Y}^P \setminus c_1} h_{c_2}(x)\right)\right],$$

where $\mathfrak{d}(c_1,c_2)$ is a semantic distance between the concepts c_1 and c_2, *e.g.*, a graph distance d_G (section 6.1.2.2). We hypothesize that this formulation improves upon the performance of the conventional 1-vs-2 method.

8. Conclusion

This chapter concludes the thesis by offering an overarching summary of our findings. We also point towards promising future work opportunities that arise from open questions and interesting combinations of methods.

8.1. Summary

In chapter 1, we argue that methodical improvements such as better neural network architectures or optimization algorithms do not necessarily apply to any given machine learning task. Formally, this is a consequence of the *no free lunch* theorem (see section 2.1.1.5), and is a serious concern for applications where labeled training data is scarce and expensive, *e.g.*, biodiversity monitoring (see section 7.3.2). We then identify possible contributions to other aspects of a learning system that are not subject to this fundamental limit. These two "escape routes" concern the training data on one hand and the learning paradigm on the other hand. We further consider properties of real-world applications (see section 1.1) to focus our contributions.

First Escape Route — Training Data

Concept hierarchies (see section 3.2) contain knowledge in the form of relational statements, *e.g.*, "a bird is an animal". We replace the classes in a typical classification tasks with concepts (see section 3.1.1) from such a hierarchy, which are no longer mutually exclusive. The hierarchy further allows for a distinction of precise and imprecise concepts (see section 5.1). Precise concepts, *e.g.*, `snow bunting`, do not subsume any other concepts. They are the leaf nodes in a graph representation, and correspond to the original classes. Imprecise concepts, *e.g.*, `object` or `animal`, are the inner nodes, and are not present in a typical classification task.

Knowledge Integration In section 5.2, we introduce a hierarchical classifier that predicts probabilities for all concepts, precise and imprecise. The probabilistic model is constrained by the hierarchy, such that the probability of a subsumed concept is at most equal to the probability of the subsuming concept (see section 5.2.2). We only allow precise predictions. With this restriction, the hierarchical classifier is a transparent drop-in replacement for a conventional one-hot softmax classifier. These constraints integrate the knowledge contained in the concept hierarchy into the model, and if the knowledge is correct and helpful, should improve the accuracy without additional training data. The domain knowledge provides a task-specific inductive bias (see section 2.1.1.5).

We validate this method on multiple computational scales of interest in section 6.3 and the findings are discussed in detail in section 6.7.1.1. On small-scale setups with benchmark data, our method shows a substantial increase in both accuracy and training speed. However, in a large-scale investigation, we observe a small decrease in accuracy. We identify two possible causes, namely weak visual-semantic correspondence and faulty concept hierarchies, which we investigate in detail in section 6.6.1 and section 6.6.2, respectively.

Learning from Imprecise Data Still, knowledge integration is only one application of concept hierarchies. We propose learning from imprecise data in section 5.1 to address the real-world nature of annotators with different levels of expertise (see section 1.1). In this setting, annotators are allowed to use imprecise concepts as labels to express their uncertainty. If they were forced to select only precise labels, they would either not label the image in question at all, or guess and possibly make a mistake. Furthermore, when crawling the web for training data using a search engine, imprecise concepts offer a larger variety of search terms, and potentially more results per search as terms become more generic (see fig. 6.7). Our method CHILLAX (see section 5.3.2) can learn from training data labeled as either precise or imprecise concepts while still providing precise predictions for easier adaptation of existing tasks. We refer to this additional generalization step as extrapolation, and also propose performing extrapolation during training in section 5.3.3, thereby generating pseudo-labels to further improve performance.

In section 6.4, we conduct a large-scale study of learning from imprecise data on benchmark datasets, where artificially introduce imprecision to simulate different data sources, *e.g.*, volunteer annotators or web crawling. These imprecision models are described in section 5.1.1.1 and validated on real-world data in section 6.6.3. Compared to two baselines that represent an annotator guessing or not labeling when they are uncertain, CHILLAX performs substantially better. We also observe that it is reasonably robust to label noise on top of imprecision and show that it has distinct advantages over its competitor HEX (Deng et al. 2014).

Analyses While we can use concept hierarchies to structure our models, this process can only improve the models if the knowledge represented by the hierarchies is correct and relevant. Because our applications are all computer vision tasks, *i.e.*, visual problems, it is not immediately clear where semantic knowledge could be helpful and in which way the two domains are related. In section 6.6.1, we directly measure the correlation between semantic and visual measures (see section 6.1.2.2) on pairs of images and concepts. We show that the correlation between visual and semantic similarity is stronger when using the concept hierarchy as opposed to a "flat" hierarchy. The findings are discussed further in section 6.7.3.1.

Even without the visual-semantic domain gap, a concept hierarchy could contain false statements, *e.g.*, "a bird is a vehicle". In section 6.6.2, we introduce errors into a hierarchy in a controlled manner on benchmark data and observe a negative effect on accuracy. We repeat the experiment on D-CHIVES, a synthetic dataset with perfect visual-semantic correspondence (see section 6.6.2.2), and find that the negative effect is substantially stronger. In section 6.7.3.2, we discuss possible causes of this discrepancy.

Real-World Data We contemplate several potential sources of imprecise data in section 5.1.1.1. Web crawling is one possible data source, where websites are scanned systematically to produce pairs of images and captions. These pairs can be transformed into training data for classification tasks, *e.g.*, using the algorithm in section 5.1.2. Alternatively, one can determine the relevant concepts in advance and use an image search to produce labeled data. Here, imprecise data allows for more search terms.

We test a webly supervised learning setup in section 6.5, where we augment a real-world biodiversity monitoring dataset with images downloaded from Flickr and use a concept hierarchy composed using WikiSpecies queries to determine the search terms. This example illustrates the relative ease of obtaining a concept hierarchy, even for a very specific domain. Our hierarchical classifier already has an advantage on the initial dataset, and CHILLAX outperforms the baseline on the combined data as well. An off-label use of our hierarchical classifier as a filter further improves accuracy. The results are laid out and discussed more in section 6.7.2.

Our experiments in section 6.4 rely heavily on the noise models proposed in section 5.1.1.1. Hence, it is crucial that we validate the models to support the conclusions of these experiments. In section 6.6.3, we analyze a whole year of Flickr uploads instead of focusing on specific search terms. We map individual metadata elements (title, tags etc.) to concepts using the algorithm in section 5.1.2 and observe the distributions over depth in the hierarchy. The results show that except for a quirk of the WordNet hierarchy, the real-world distributions fit the model reasonably well. We further interpret the results in section 6.7.3.3.

Second Escape Route — Learning Paradigm

Lifelong learning (see section 7.1) addresses further differences between the typical academic setup of "waterfall" learning and the needs of real-world applications (see section 1.1). It focuses on the distribution of project resources over time, *i.e.*, annotator sessions and image acquisition. We use camera trap image analysis as a recurring example of a real-world application with these properties.

Active and Incremental Learning for Object Detection A lifelong learning systems has two main methodical ingredients in active learning (see section 7.1.1) and incremental learning (see section 7.1.2). In section 7.2.1, we derive active learning value functions for object detection tasks where we consider the value of whole images as opposed to instances. We propose aggregated classification value functions in section 7.2.1.1 and two metrics specific to the YOLO object detector (Redmon et al. 2016). To complete the lifelong learning system, we adapt the deep fine-tuning method from Käding et al. (2016b) to YOLO in section 7.2.2.

In section 7.2.3, we validate the system on benchmark data. It does not exhibit substantial amounts of catastrophic forgetting, which points to the reliability of the incremental learning method. The active learning methods that compute sums of the values of instances perform the best overall, likely due to their preference of examples with many detections. This preference can be observed qualitatively in section 7.2.3.4 and the results are discussed further in section 7.2.4.

Carpe Diem — Weakly Supervised Lifelong Object Detection To evaluate the active and incremental learning methods in a real-world setting, we develop a graphical user interface called *Carpe Diem* (in reference to YOLO). The user experience and flow of interactions is detailed in section 7.3.4. It integrates the annotation method of Papadopoulos et al. (2016), where users can only confirm or reject a proposed bounding box instead of drawing them manually.

Researchers are asked to annotate several batches of camera trap images to validate the entire system in section 7.3.3. We observe a substantial accuracy increase over the initial model even after the first batch. The active learning component successfully selects valuable images from a dataset where almost two thirds of the images are spuriously triggered and empty.

Whole Image Annotations and Face Detection for Re-Identification Any labeling process should be as flexible and as accommodating towards annotators as possible (see section 1.1). In section 7.4, we perform individual classification of *gorilla gorilla gorilla* by extracting faces first. Images are only annotated with a single label and no bounding boxes. Such simple annotations can comfortably and efficiently be collected in the field during image acquisition.

The images can contain multiple individuals, in which case the annotators consider the individual in focus. We train a face detector separately from a classifier that only processes extracted faces. With this process, we can utilize the whole image annotations, but still leverage strong facial features. When the detector predicts more than one face, we use heuristics to determine which individual is in focus of the image. The experiments in section 7.4.2 show that this separation of concerns is successful despite the accumulation of errors from detector and classifier.

8.2. Future Work

In this section, we identify promising future research directions that continue the work presented in this thesis. The directions include remaining open questions as well as new ideas.

Interaction Our motivation for developing methods around imprecise data is that it allows annotators to express their uncertainty or lack of expertise correctly. In this thesis, we focus on the processing of the resulting imprecise data. However, human-machine interaction also affects the efficiency of a learning system (see section 7.3), and the optimal interaction process for generating imprecise labels is not clear.

A top-down approach, where an annotator starts from the root and navigates through the hierarchy with increasing precision, could be well suited when there are numerous concepts. However, it requires many steps for “deep” hierarchies. Conversely, in a bottom-up approach, the annotator would select multiple concepts that probably apply to the image in question, and the label is determined by the lowest common subsumer (see section 6.1.2.2). The bottom-up approach would be best suited to a small hierarchy. Both approaches could be improved by incorporating predictions from an already trained model to pre-select relevant concepts and “skip” levels in the concept hierarchy.

Probabilistic Connections Our hierarchical classifier is based on the subsumption assumption (see section 5.2.2). In fact, it plays such a central role that no prediction of the classifier can possibly violate it. Still, we learn in sections 6.6.1, 6.6.2 and 6.7.3 that there are many reasons why it might not always hold in practice.

To resolve this discrepancy, we propose to regard each individual element of the hyponymy relation as probabilistic. For example, this probability could be fixed and equal for the whole relation, adding a global relaxation term to the classifier. It could also be learned by the hierarchical classifier, which would allow for interesting insight by studying the distribution over the concept hierarchy.

The setup could further be generalized by also allowing new connections to be learned. The original concept hierarchy would only be used to initialize (and limit) a connection matrix, similar to the affinity matrix of Fergus et al. (2010). This matrix is then refined by the classifier as part of the learning process. Further regularization would be required to ensure that the matrix actually describes a hierarchical relation (see section 3.2.1).

Multiple Hierarchies and Parts of Speech In section 3.3, we list several possible sources of concept hierarchies. Furthermore, we learn in section 5.1.2 that mapping labels in a dataset to concepts in a hierarchy is not necessarily uniquely defined. Still, the classifier only considers one "true" set of concepts and a single relation. While probabilistic connections could resolve this discrepancy, there is an alternative approach, namely allowing more than one hierarchy.

Such an approach is proposed in Li et al. (2021) and our hierarchical classifier could similarly be adapted to consider multiple, possibly overlapping, sets of imprecise concepts. Moreover, this modification would allow for inclusion of different parts of speech at the same time.

A set of attributes, some of which are shared by different concepts, is semantically distinct from a hyponymy relation. However, it would fit well with the assumptions in section 5.2.2 if a global attribute that matches all other attributes and concepts is introduced to preserve the closed world. Using multiple hierarchies at the same time, attributes and hyponymy could be combined similar to USE (Hwang and Sigal 2014, see also section 4.1.2). But unlike USE, this approach would not be limited to two semantic relations. For example, one could also include antonymy or meronymy (see section 3.1.2.2).

Lifelong Learning with Imprecise Data In section 7.5, we propose an active learning value function for classification that considers the semantic distance between predicted concepts. Otherwise, an example where `cat` and `dog` are predicted with equal confidence would be treated the same as an example where `cat` and `bus` are equally likely. However, *hierarchical 1-vs-2* is only designed as a replacement to improve conventional *1-vs-2* in regular classification tasks where only precise concepts occur, and there is no consideration of imprecise data.

We pose that imprecise data brings further opportunities for changes in the learning paradigm. For example, an active learning method could not only assign a value to a certain example, but also request an annotation with only limited precision to conserve a small annotation budget, assuming that less precise labels can be acquired for less cost. Furthermore, it could spend a large fraction of the budget on a single example to obtain a maximally precise annotation, possibly from multiple annotators, if the example is considered valuable enough.

Incremental learning requires additional considerations for hierarchical classification as well, for example, when new concepts are introduced. Our implementation CHIA (see section 1.2) can perform incremental learning of our hierarchical classifier and also discover new relations whenever concepts are added. Such an approach could be combined with a hierarchy-informed curriculum, *e.g.*, as proposed in Stretcu et al. (2021) (see section 4.1.1.3). However, this method would have to be adapted to consider the order of precision in the training data as well, instead of only focusing on the classifier.

A. Appendix

Table A.1.: Hyperparameters and further setup details of small-scale experiments on benchmark datasets.

	CIFAR-100	NABirds	ImageNet-1k
Optimization			
Algorithm	Adam (see section 2.1.3.3)		
Learning Rate	0.00		
Minibatch Size	128	64	
Iterations	60 000	120 000	234 274
Architecture	ResNet-32	ResNet-50 V2	
Input Size	32×32	224×224	
Initialization	Random, eq. (2.12)		
Regularization	L2, 0.00	L2, 0.00	
Processing and Augmentation (see section 6.2.3)			
Random Shifts	Up to 4 px	Up to 32 px	
Random Flips	Horizontal only		
Processing	Scaling to $[0, 1]$ only		
Experimental Runs	6	3	

Table A.2.: Hyperparameters and further setup details of large-scale experiments on benchmark datasets.

	NABirds	ImageNet-1k
Optimization		
Algorithm	SGD (see section 2.1.3.1), $\alpha = 0.9$	
Learning Rate Schedule	SGDR (see section 6.2.2.3)	
	$T_0 = 59760, \eta = 0.003$	$T_0 = 100091, \eta = 0.2$
Minibatch Size	32	128
Iterations	59 760	200 000
Architecture	ResNet-50 V2	
Input Size	448 × 448	224 × 224
Initialization	iNaturalist 2017	Random, eq. (2.12)
Regularization	None	
Processing and Augmentation (see section 6.2.3)		
Random Flips	Horizontal only	
Random Crops	Out of 512 × 512	Out of 256 × 256
Processing	Mean and std. dev. normalization (see section 6.2.3.2)	
Experimental Runs	6	1

Table A.3.: Hyperparameters and further setup details of large-scale experiments on benchmark datasets with imprecise data. Table compiled of data from Brust, Barz, and Denzler (2021b).

	NABirds	ImageNet-1k
Optimization		
Algorithm	SGD (see section 2.1.3.1), $\alpha = 0.9$	
Learning Rate Schedule	SGDR (see section 6.2.2.3)	
	$T_0 = 59760, \eta = 0.0044$ (CHILLAX) $T_0 = 59760, \eta = 0.003$ (baselines)	$T_0 = 100091, \eta = 0.2$
Minibatch Size	32	128
Iterations	59 760	200 000
Architecture	ResNet-50 V2	
Input Size	448 × 448	224 × 224
Initialization	iNaturalist 2017	Random, eq. (2.12)
Regularization	None	
Processing and Augmentation (see section 6.2.3)		
Random Flips	Horizontal only	
Random Crops	Out of 512 × 512	Out of 256 × 256
Processing	Mean and std. dev. normalization (see section 6.2.3.2)	
Experimental Runs	6	1

Table A.4.: Large-scale benchmark comparison of CHILLAX and baselines on NABirds. We report the accuracy in percent on the validation set. Imprecision as indicated, 10 % inaccuracy. Table compiled of data from Brust, Barz, and Denzler (2021b).

Deng et al. p	0.99	0.95	0.9	0.5
Leaves only	8.19 ± 0.57	23.47 ± 1.11	39.15 ± 1.60	70.48 ± 0.73
Random leaf	57.16 ± 0.04	57.65 ± 0.10	59.17 ± 0.14	69.20 ± 0.37
CHILLAX	**58.34** ± 0.70	**63.61** ± 0.50	**67.76** ± 0.35	**74.26** ± 0.50
Geometric q	0.5	0.8	0.9	0.95
Leaves only	36.36 ± 1.01	**69.17** ± 0.11	**73.75** ± 0.24	**75.78** ± 0.34
Random leaf	9.44 ± 0.17	46.58 ± 0.60	62.65 ± 0.15	69.91 ± 0.16
CHILLAX	**38.70** ± 0.70	67.35 ± 0.19	72.07 ± 0.15	73.48 ± 0.23
Poisson λ	1.0	2.0	3.0	4.0
Leaves only	22.12 ± 0.40	54.38 ± 1.18	67.95 ± 0.07	**73.06** ± 0.64
Random leaf	9.99 ± 0.33	33.13 ± 0.57	53.10 ± 0.74	65.44 ± 0.21
CHILLAX	**34.61** ± 1.20	**60.50** ± 0.33	**69.84** ± 0.26	72.81 ± 0.16

Table A.5.: Large-scale benchmark comparison of CHILLAX and adaptive threshold method on NABirds. We report the accuracy in percent on the validation set. Imprecision as indicated, no inaccuracy. Table compiled of data from Brust, Barz, and Denzler (2022).

Noise model	(ii)	(iii)	(iv)	(v)
Do nothing	62.66 ± 0.82	49.04 ± 1.04	43.18 ± 0.20	70.91 ± 0.34
Best non-adaptive	**63.54** ± 0.51	**50.68** ± 0.44	**44.02** ± 0.12	71.77 ± 0.00
Adaptive threshold:				
$\delta\mathfrak{I}^* = 0.025$	61.80 ± 0.69	49.07 ± 0.93	43.35 ± 0.82	71.58 ± 0.40
$\delta\mathfrak{I}^* = 0.0375$	61.43 ± 0.51	49.75 ± 1.13	43.11 ± 1.49	72.00 ± 0.43
$\delta\mathfrak{I}^* = 0.05$	61.80 ± 0.44	49.52 ± 0.90	42.92 ± 0.42	**72.10** ± 0.31
$\delta\mathfrak{I}^* = 0.0625$	61.87 ± 0.91	49.57 ± 1.55	42.30 ± 0.35	71.15 ± 0.92
$\delta\mathfrak{I}^* = 0.075$	62.20 ± 0.76	49.97 ± 0.59	41.71 ± 1.13	71.37 ± 0.40
$\delta\mathfrak{I}^* = 0.1$	61.79 ± 0.30	48.57 ± 0.61	39.72 ± 0.96	70.77 ± 0.34
$\delta\mathfrak{I}^* = 0.15$	61.70 ± 0.46	44.53 ± 1.56	33.56 ± 1.16	69.02 ± 0.87
$\delta\mathfrak{I}^* = 0.2$	61.68 ± 0.56	41.04 ± 0.51	30.42 ± 0.46	68.56 ± 0.46
$\delta\mathfrak{I}^* = 0.3$	61.34 ± 0.35	34.03 ± 1.04	27.30 ± 1.11	66.31 ± 0.52
$\delta\mathfrak{I}^* = 0.4$	61.98 ± 0.55	30.49 ± 0.67	25.02 ± 0.78	64.64 ± 1.06
$\delta\mathfrak{I}^* = 0.5$	61.69 ± 0.40	26.52 ± 1.28	23.72 ± 0.53	65.04 ± 1.04

Table A.6.: Realized IC gain of self-supervised CHILLAX methods on NABirds. Imprecision as indicated, no inaccuracy.

Noise model	(ii)	(iii)	(iv)	(v)
Adaptive threshold:				
$\delta\mathfrak{I}^* = 0.025$	0.03	0.04	0.04	0.04
$\delta\mathfrak{I}^* = 0.0375$	0.04	0.05	0.05	0.06
$\delta\mathfrak{I}^* = 0.05$	0.05	0.07	0.07	0.07
$\delta\mathfrak{I}^* = 0.0625$	0.06	0.08	0.08	0.09
$\delta\mathfrak{I}^* = 0.075$	0.07	0.10	0.10	0.10
$\delta\mathfrak{I}^* = 0.1$	0.07	0.13	0.13	0.13
$\delta\mathfrak{I}^* = 0.15$	0.07	0.18	0.18	0.17
$\delta\mathfrak{I}^* = 0.2$	0.07	0.23	0.23	0.21
$\delta\mathfrak{I}^* = 0.3$	0.07	0.32	0.32	0.30
$\delta\mathfrak{I}^* = 0.4$	0.07	0.41	0.41	0.37
$\delta\mathfrak{I}^* = 0.5$	0.07	0.50	0.50	0.39
IC range $[0.0, 0.2]$	0.07	0.01	0.02	0.03
IC range $[0.1, 0.3]$	0.07	0.16	0.15	0.10
IC range $[0.2, 0.4]$	0.00	0.21	0.21	0.15
IC range $[0.3, 0.5]$	0.00	0.28	0.26	0.17
IC range $[0.4, 0.6]$	0.00	0.34	0.35	0.18

Table A.7.: The 9 species present in the AMMOD moth dataset are mapped to WikiSpecies as follows.

- `amphipyra pyramidea -- berberea` → genus *Amphipyra*
- `aplocera plagiata -- efformata` → genus *Aplocera*
- `chlroclystis v-ata` → species *Chloroclystis v-ata*
- `epirrita autumnata -- dilutata -- christyi` → genus *Epirrita*
- `mesapamea spec` → genus *Mesapamea*
- `noctua janthina -- janthe` → custom concept (*Noctua comes* also exists)
- `oligia latruncula -- strigilis -- versicolor` → custom concept (other *Oligia* exist)
- `sunira circellaris` → subgenus *Sunira*
- `thera variata -- britannica` → genus *Thera*

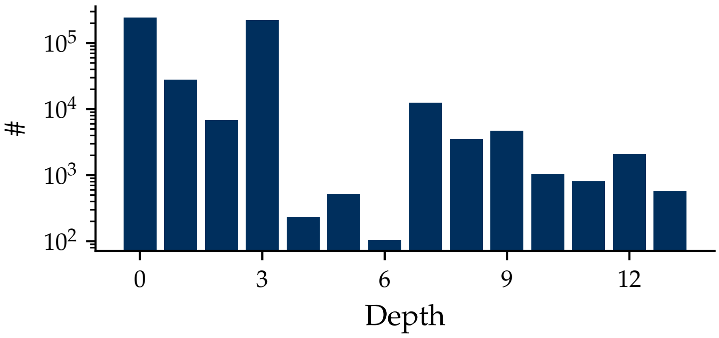
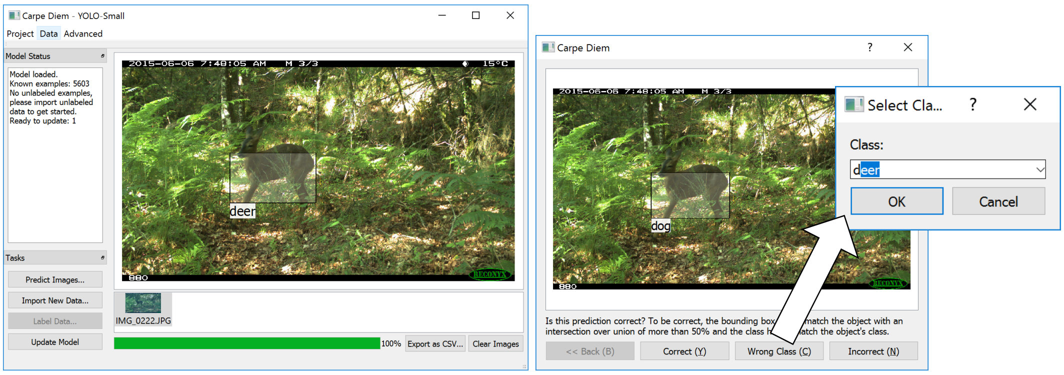


Figure A.1.: Average number of Flickr search results per moth taxon at depth in the concept hierarchy.

Table A.8.: Face detection performance on Mbeli Bai held-out validation set. Table compiled of data from Brust et al. (2017).

# Training Images	—	500	1000	1500	2000
Validation set					
AP (%)	29.5	86.6	88.4	90.6	**90.8**
Precision (%)	83.1	85.6	89.4	88.9	90.1
Recall (%)	31.5	88.7	90.3	92.0	92.2
Whole dataset					
No Detection (%)	59.8	0.7	0.8	0.3	0.4
1 Detection (%)	39.1	92.3	93.7	93.9	**95.4**
>1 Detection (%)	1.1	7.0	5.4	5.9	4.1



Model loaded.
Known examples: 5603
No unlabeled examples, please import unlabeled data to get started.
Ready to update: 1

Class: deer

Is this prediction correct? To be correct, the bounding box match the object with an intersection over union of more than 50% and the class h atch the object's class.

Figure A.2.: Carpe Diem graphical user interface. Left: landing page. Right: correcting a class label. Figure taken from Brust, Käding, and Denzler (2020).

Notation

This section summarizes the notation used throughout this work. We aim for a short, expressive notation that minimizes potential for ambiguity. It seeks to be mostly compatible to Shalev-Shwartz and Ben-David (2014) as well as Goodfellow, Bengio, and Courville (2016).

Basics

a	A scalar
$\boldsymbol{a}$	A vector
$\boldsymbol{A}$	A matrix
$f(x)$	A function $f : x \mapsto f(x)$
$\mathbb{P}(A)$	Probability of event A
$\mathbb{E}(A)$	Expectation of random variable A
$\mathbb{1}[A](x)$	Indicator function of x for set A (argument can be elided when obvious)
$\mathcal{P}(A)$	The power set of set A

Learning

$\mathcal{X}$	Domain set
$\mathcal{Y}$	Label set
$x \in \mathcal{X}$	An element of the domain set (a domain point)
$y \in \mathcal{Y}$	An element of the label set
S	Training data $S = \big((x_1, y_1), \ldots, (x_m, y_m)\big)$
h	Hypothesis
$\mathcal{A}$	Learner
f	Target function
$\mathcal{D}$	Environment distribution
$L_{\mathcal{D},f}(h)$	Risk
$L_S(h)$	Empirical Risk w.r.t. training data and hypothesis
$\mathcal{L}$	Loss function
Θ	Real vector space of parameters
$\boldsymbol{\theta}$	Element of Θ
η	Learning rate
$\mathbb{I}$	The set of images
$\mathcal{Y}^{BB}$	The set of possible bounding boxes $\mathcal{Y}^{BB} = \mathbb{R}^4 = \mathbb{R}^2 \times \mathbb{R}^2$

Semantics

$\mathcal{Y}^P$	The set of labels for prediction and validation
$\mathcal{Y}^T$	The set of labels for training
`dog`	A class label
C. diff.	A species
$\mathcal{C}$	A set of concepts
R	A relation over concepts
$\leq_{\text{is-a}}$	The hyponymy relation
E	The extension of a concept
I	The intension of a concept
G	A graph with vertices and edges(V, E)
d_G	Graph distance
$\mathfrak{I}$	Information content

List of Figures

2.1. XOR problem 28

3.1. Non-mutually exclusive set of concepts 34
3.2. Graph representations of a concept hierarchy 39
3.3. Diamond pattern in graph 40
3.4. Example image from Visual Genome 43
3.5. Example image from Microsoft COCO 43
3.6. Skin lesion hierarchy 44

5.1. Imprecise and precise concepts in a hierarchy 60
5.2. Depth-based imprecision models 63
5.3. Label encoding and loss mask for precise label 72
5.4. Examples of different training, validation and prediction label sets . . 75
5.5. Loss masks for imprecise label 77
5.6. Subhypotheses for imprecise label 79

6.1. Relation of intrinsic IC and CHILLAX accuracy on NABirds 101
6.2. Examples of unsupervised filtering on Flickr moths 110
6.3. Correlation matrix of visual and semantic similarities 116
6.4. Examples of high and low visual-semantic agreement on CIFAR-100 . 117
6.5. Example images from D-CHIVES L 119
6.6. Example image with metadata from Flickr 121
6.7. Flickr images with metadata synsets at different depths in WordNet . . 122

7.1. Most and least valuable images from PASCAL VOC 2012 137
7.2. Example images from Peneda-Gerês 139
7.3. Example images from Mbeli Bai 142
7.4. Example motivating hierarchical 1-vs-2 144

A.1. Flickr images at different depths of moth taxa 155
A.2. Carpe Diem graphical user interface 155

List of Tables

6.1. Small-scale knowledge integration benchmark results 92
6.2. Large-scale knowledge integration benchmark results 92
6.3. IC and remaining precise examples in imprecise NABirds 97
6.4. IC and remaining precise examples in imprecise ImageNet-1k 97
6.5. Large-scale inaccurate data results on NABirds 98
6.6. Large-scale imprecise data results on NABirds, no inaccuracy 99
6.7. Large-scale imprecise data results on NABirds, 1 % inaccuracy 100
6.8. Large-scale inaccurate data results on ImageNet-1k 103
6.9. Large-scale imprecise data results on ImageNet-1k, no inaccuracy . . . 103
6.10. Large-scale self-supervision simulation on imprecise NABirds 104
6.11. Large-scale non-adaptive self-supervision results on imprecise NABirds 106
6.12. Moth dataset knowledge integration results 111
6.13. Moth dataset with webly supervised imprecise data results 112
6.14. Synthetic data knowledge integration results, with faulty hierarchy . . 120

7.1. Active learning object detection benchmark results on VOC 2012 . . . 136
7.2. Individual identification on Mbeli Bai 143

A.1. Small-scale knowledge integration benchmark setup 151
A.2. Small-scale knowledge integration benchmark setup 152
A.3. Large-scale imprecise data benchmark setup 152
A.4. Large-scale imprecise data results on NABirds, 10 % inaccuracy 153
A.5. Large-scale adaptive threshold results on NABirds 153
A.6. Realized IC gain of IC Range and adaptive threshold on NABirds . . . 154
A.7. Moth dataset mapping to WikiSpecies 154
A.8. Face detection on Mbeli Bai 155

Acronyms

AP	average precision.
AULC	area under the learning curve.
CNN	convolutional neural network.
DAG	directed acyclic graph.
ERM	empirical risk minimization.
HoG	histogram of oriented gradients.
IC	information content.
mAP	mean average precision.
MLP	multi-layer perceptron.
NLP	natural language processing.
OCR	optical character recognition.
OHSM	one-hot softmax.
PAC	probably approximately correct.
RDF	Resource Description Framework.
RLM	regularized loss minimization.
ROC	receiver operater characteristics.
SGD	stochastic gradient descent.
SVM	Support Vector Machine.
VQA	visual question answering.

Glossary

1-vs-2	Active learning strategy, also known as margin sampling. 130
Activation Function	Nonlinear function used in neural networks. 27
Active Learning	Methods that can select valuable samples for labeling with the highest possible efficiency. 130
Bias	Additive term in a linear model. 27
Bounding Box	An axis-aligned rectangle that describes a region of an image. 23
Classification	A supervised learning task where the label set is discrete. 8
Concept	A more general term for a class, relaxing the requirement of mutual exclusivity. 33
Concept Hierarchy	Set of concepts with structure given by a semantic relation. 37
Context	Structure upon which concepts are defined. 34
Convolutional Layer	Layer of neurons where each is connected to a neighborhood of input neurons and weights are shared. 29
Convolutional Neural Network	Type of neural network used in deep learning. 26, 163
Data Augmentation	Set of transforms on training data that exploit invariances to increase the sample size. 89
Deep Learning	An approach to machine learning where features are learned from large amounts of data instead of being defined by hand. 26
Domain Point	An element of the domain set. 8
Domain Set	The “input” portion of a supervised learning problem. 8
Empirical Risk	Error measure for a given hypothesis and training data. 9
Environment Distribution	The distribution that the training data is sampled from. 9
Example	An element of the task-agnostic example set of training data. 12

Example Set	A task-agnostic way of describing the set from which training data is sampled. 12
Feature	A representation of data with certain properties that make it useful for machine learning applications. 25
Feature Map	Part of the output of a convolutional layer. 30
Fine-grained Recognition	A classification task where classes are closely related in some fashion. 23
Fully-connected Layer	Layer of neurons where each is connected to all input neurons. 27
Generalized Loss Function	A measure of success for a general learning problem. 13
Gradient Descent	A first-order optimization algorithm. 14
Hierarchical Classification	Classification task that involves a concept hierarchy. 67
Hyperparameter	Parameter that controls optimization, but is typically not itself optimized using training data. 15
Hyponymy	A hyponym is a concept subsumed by a more general hypernym, and hyponymy is the corresponding relation between concepts. 36
Hypothesis	A function from the domain set to the label set — the output of a learner. 8
Image	A discretized representation of the image function. 22
Image Classification	A supervised learning task on the domain of images where the label set is discrete. 23
Image Function	A continuous representation of an image, before discretization. 21
Imprecise Data	Training data that contains semantically imprecise labels. 60
Incremental Learning	Methods that can add new knowledge to an existing model without causing it to forget previous data. 131
Inductive Bias	Prior knowledge about a task that can be used to draw further conclusion from training data. 10
Instance	A single object, represented by a bounding box, in the context of object detection. 23
Internal Label	Label representation that is used by a method internally only exposed via a translation process. 68
Internal Label Set	Set of internal labels. 68
Kernel	Coefficients of a convolution operation. 29

Label	An element of the label set. 8
Label Set	The "output" portion of a supervised learning problem. 8
Layer	Collection of neurons, organizational unit in a neural network. 27
Learner	An algorithm that produces hypotheses from training data. 9
Learning Rate	Hyperparameter in gradient-based optimization methods. 15
Lifelong Learning	Combination of active learning and incremental learning in a continuous cycle. 129
Linear Model	A hypothesis composed partly of a linear function. 13
Loss Function	A measure of success for a prediction problem. 12
Machine Learning	A field related to artificial intelligence where predictions are made based on a mathematical model created from training data. 7
Multi-layer Perceptron	Multiple perceptrons arranged into layers to form a complex neural network. 27, 163
Neural Network	Machine learning method inspired by construction patterns found in the human brain. 26
Novelty Detection	Machine learning task where examples of previously unseen concepts are detected. 136
Object Detection	A supervised learning task where objects in an image are localized by bounding boxes and classifier. 23
Perceptron	Mathematical model of a single neuron, the building block of neural networks. 27
Pixel	A single element of a discretized image. 23
Pseudo-label	A label that is not annotated by a human, but automatically generated in the context of self-supervised learning. 77
Representation Learning	Learning of a feature representation, as opposed do hand-engineered features. 26
Risk	Error measure for a given hypothesis, environment distribution and target function. 9
Sample Complexity	The number of training data samples required to achieve a certain goal. 11
Self-supervised Learning	The use of confident predictions as labels. 16

Semantic Segmentation	A supervised learning task where each pixel of an image is classified individually. 23
Semi-supervised Learning	A learning problem where both unlabeled and labeled training data is available. 16
Statistical Learning Theory	A mathematical framework relating probability theory and machine learning. 7
Structured Output Prediction	A machine learning task with a very large label set endowed with a structure. 34
Supervised Learning	A learning problem where predictions relating “inputs” to “outputs” are made by learning from labeled training data. 16
Support Vector Machine	A linear classifier. 14, 163
Target Function	A function that labels every element of the domain set “correctly”. 9
Task	Formal definition of a machine learning problem and its constituent components. 13
Taxonomy	Set of all concepts and their extensions given a context. 35
Training Data	A sequence of pairs relating elements of the domain set to elements of the label set. 8
Unsupervised Learning	A learning problem where patterns are discovered using only unlabeled training data. 16
Validation Set	Data sampled from the sample environment distribution as training data, but independent. 20
Weakly Supervised Learning	Supervision mode between supervised learning and unsupervised learning, where the quality of labels is reduced. 17
Weight	Coefficient in a linear model. 27

References

Ahmed, Karim, Mohammad Haris Baig, and Lorenzo Torresani (2016). “Network of Experts for Large-scale Image Categorization”. In: *European Conference on Computer Vision (ECCV)*. DOI: `10.1007/978-3-319-46478-7_32`.

Alberts, Houda, Teresa Huang, Yash Deshpande, Yibo Liu, Kyunghyun Cho, Clara Vania, and Iacer Calixto (2020). “VisualSem: A High-quality Knowledge Graph for Vision and Language”. In: arXiv: `2008.09150v1 [cs.CL]`.

AlMousa, Mohannad, Rachid Benlamri, and Richard Khoury (2021). “A Novel Word Sense Disambiguation Approach Using Wordnet Knowledge Graph”. In: arXiv: `2101.02875v1 [cs.CL]`.

Amstrup, Steven C, Trent L McDonald, and Bryan FJ Manly (2010). *Handbook of Capture-recapture Analysis*. Princeton University Press. ISBN: 978-0-691-08967-6.

Antol, Stanislaw, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh (2015). “VQA: Visual Question Answering”. In: *International Conference on Computer Vision (ICCV)*. DOI: `10.1109/ICCV.2015.279`.

Arponen, Heikki and Tom E. Bishop (2019). “SHREWD: Semantic Hierarchy-based Relational Embeddings for Weakly-supervised Deep Hashing”. In: *International Conference on Learning Representations Workshops (ICLR-WS)*. arXiv: `1908.05602v1 [cs.IR]`.

Barata, Catarina, Jorge S. Marques, and M. Emre Celebi (2019). “Deep Attention Model for the Hierarchical Diagnosis of Skin Lesions”. In: *Conference on Computer Vision and Pattern Recognition Workshops (CVPR-WS)*. DOI: `10.1109/CVPRW.2019.00334`.

Barz, Björn and Joachim Denzler (2019). “Hierarchy-based Image Embeddings for Semantic Image Retrieval”. In: *Winter Conference on Applications of Computer Vision (WACV)*. DOI: `10.1109/WACV.2019.00073`.

Barz, Björn and Joachim Denzler (2021a). “Content-based Image Retrieval and the Semantic Gap in the Deep Learning Era”. In: *International Conference on Pattern Recognition Workshops (ICPR-WS)*. DOI: `10.1007/978-3-030-68790-8_20`.

Barz, Björn and Joachim Denzler (2021b). “WikiChurches: A Fine-grained Dataset of Architectural Styles with Real-world Challenges”. In: arXiv: `2108.06959 [cs.CV]`.

Beckwith, Richard and George A. Miller (1990). "Implementing a Lexical Network". In: *International Journal of Lexicography* 3 (4), pp. 302–312. DOI: `10.1093/ijl/3.4.302`.

Beery, Sara, Elijah Cole, and Arvi Gjoka (2020). "The iWildCam 2020 Competition Dataset". In: arXiv: `2004.10340 [cs.CV]`.

Bellman, Richard (1957). *Dynamic Programming*. Princeton University Press. ISBN: 978-0691079516.

Benkhalifa, Mohammed, Abdelhak Mouradi, and Houssaine Bouyakhf (2001). "Integrating WordNet Knowledge to Supplement Training Data in Semi-supervised Agglomerative Hierarchical Clustering for Text Categorization". In: *International Journal of Intelligent Systems* 16.8, pp. 929–947. DOI: `10.1002/int.1042`.

Bertinetto, Luca, Romain Mueller, Konstantinos Tertikas, Sina Samangooei, and Nicholas A. Lord (2020). "Making Better Mistakes: Leveraging Class Hierarchies with Deep Networks". In: *Computer Vision and Pattern Recognition (CVPR)*. DOI: `10.1109/CVPR42600.2020.01252`.

Bilal, A., A. Jourabloo, M. Ye, X. Liu, and L. Ren (2018). "Do Convolutional Neural Networks Learn Class Hierarchy?" In: *IEEE Transactions on Visualization and Computer Graphics*. DOI: `10.1109/TVCG.2017.2744683`.

BIPM, IEC, IFCC, ILAC, IUPAC, IUPAP, ISO, and OIML (2012). *The International Vocabulary of Metrology. Basic and General Concepts and Associated Terms*. Ed. by JCGM. 3rd ed. URL: `http://www.bipm.org/vim` (visited on 02/23/2021).

Birodkar, Vighnesh, Hossein Mobahi, and Samy Bengio (2019). "Semantic Redundancies in Image-classification Datasets: The 10% You Don't Need". In: arXiv: `1901.11409v1 [cs.CV]`.

Bishop, Christopher (2008). *Pattern Recognition and Machine Learning*. Springer. ISBN: 978-0-387-31073-2.

Bodesheim, Paul, Alexander Freytag, Erik Rodner, Michael Kemmler, and Joachim Denzler (2013). "Kernel Null Space Methods for Novelty Detection". In: *Computer Vision and Pattern Recognition (CVPR)*. DOI: `10.1109/CVPR.2013.433`.

Böhlke, Julia, Dimitri Korsch, Paul Bodesheim, and Joachim Denzler (2021). "Lightweight Filtering of Noisy Web Data: Augmenting Fine-grained Datasets with Selected Internet Images". In: *Computer Vision Theory and Applications (VISAPP)*. URL: `https://pub.inf-cv.uni-jena.de/pdf/Julia21:NoisyWeb` (visited on 07/02/2021).

Brickley, Dan and R.V. Guha, eds. (Feb. 25, 2014). *RDF Schema*. W3C Recommendation. Version 1.1. World Wide Web Consortium. URL: `https://www.w3.org/TR/rdf-schema/` (visited on 02/09/2021).

Brinton, Laurel J. (2000). *The Structure of Modern English. A Linguistic Introduction*. John Benjamins Publishing. ISBN: 978-902722567-2. DOI: `10.1075/z.94`.

Cardinale, Bradley J, J Emmett Duffy, Andrew Gonzalez, David U Hooper, Charles Perrings, Patrick Venail, Anita Narwani, Georgina M Mace, David Tilman, David A Wardle, et al. (2012). "Biodiversity Loss and Its Impact on Humanity". In: *Nature* 486.7401, pp. 59–67. DOI: `10.1038/nature11148`.

Cauchy, Augustin (1847). "Méthode Générale Pour La Résolution Des Systemes D'équations Simultanées". In: *Comptes rendus de l'Académie des sciences* 25.1847, pp. 536–538.

Chang, Chih-Chung and Chih-Jen Lin (2011). "LIBSVM: A Library for Support Vector Machines". In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 2.3. DOI: `10.1145/1961189.1961199`.

Chang, Dongliang, Kaiyue Pang, Yixiao Zheng, Zhanyu Ma, Yi-Zhe Song, and Jun Guo (2021). "Your 'flamingo' Is My 'bird': Fine-grained, or Not". In: *Computer Vision and Pattern Recognition (CVPR)*. arXiv: `2011.09040v3 [cs.CV]`.

Chen, Hao-Yun, Li-Huang Tsai, Shih-Chieh Chang, Jia-Yu Pan, Yu-Ting Chen, Wei Wei, and Da-Cheng Juan (2019). "Learning with Hierarchical Complement Objective". In: arXiv: `1911.07257v1 [cs.CV]`.

Chen, Liang-Chieh, Alexander Schwing, Alan Yuille, and Raquel Urtasun (2015a). "Learning Deep Structured Models". In: *International Conference on Machine Learning (ICML)*.

Chen, Tianshui, Wenxi Wu, Yuefang Gao, Le Dong, Xiaonan Luo, and Liang Lin (2018). "Fine-grained Representation Learning and Recognition by Exploiting Hierarchical Semantic Embedding". In: *ACM International Conference on Multimedia (MM)*. DOI: `10.1145/3240508.3240523`.

Chen, Xinlei, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollar, and C. Lawrence Zitnick (2015b). "Microsoft Coco Captions: Data Collection and Evaluation Server". In: arXiv: `1504.00325v2 [cs.CV]`.

Cheng, Lele, Xiangzeng Zhou, Liming Zhao, Dangwei Li, Hong Shang, Yun Zheng, Pan Pan, and Yinghui Xu (2020). "Weakly Supervised Learning with Side Information for Noisy Labeled Images". In: *European Conference on Computer Vision (ECCV)*. DOI: `10.1007/978-3-030-58577-8_19`.

Chiarcos, Christian, Sebastian Hellmann, and Sebastian Nordhoff (2011). "Towards a Linguistic Linked Open Data Cloud. The Open Linguistics Working Group". In: *Traitement automatique des langues (TAL)* 52.3, pp. 245–275.

Collell Talleda, Guillem (2016). "Is an Image Worth More Than a Thousand Words? on the Fine-grain Semantic Differences between Visual and Linguistic Representations". In: *International Conference on Computational Linguistics (COLING)*.

Cruse, D. Alan (2002). "Hyponymy and Its Varieties". In: *The Semantics of Relationships. An Interdisciplinary Perspective*. Ed. by Rebecca Green, Carol A. Bean, and Sung Hyon Myaeng. Springer-Verlag Dordrecht, pp. 3–21. ISBN: 978-90-481-5996-3. DOI: `10.1007/978-94-017-0073-3_1`.

Cui, Y., Y. Song, C. Sun, A. Howard, and S. Belongie (2018). "Large Scale Fine-grained Categorization and Domain-specific Transfer Learning". In: *Computer Vision and Pattern Recognition (CVPR)*. DOI: `10.1109/CVPR.2018.00432`.

Dalal, Navneet and Bill Triggs (2005). "Histograms of Oriented Gradients for Human Detection". In: *Computer Vision and Pattern Recognition (CVPR)*. DOI: `10.1109/CVPR.2005.177`.

Damen, Dima and Michael Wray (2020). "Supervision Levels Scale (SLS)". In: arXiv: `2008.09890v1 [cs,CV]`.

Data Science Process Alliance (2021). *What Is Waterfall?* URL: `https://www.datascience-pm.com/waterfall/` (visited on 08/18/2021).

Davis, Jim, Tong Liang, James Enouen, and Roman Ilin (2021). "Hierarchical Classification with Confidence Using Generalized Logits". In: *International Conference on Pattern Recognition (ICPR) 2020*.

Dawson, Glenn and Robi Polikar (2021). "Rethinking Noisy Label Models: Labeler-dependent Noise with Adversarial Awareness". In: arXiv: `2105.14083 [cs.LG]`.

Dehghani, Mostafa, Aliaksei Severyn, Sascha Rothe, and Jaap Kamps (2017). "Learning to Learn from Weak Supervision by Full Supervision". In: *NIPS Workshop on Meta-Learning (MetaLearn 2017)*. arXiv: `1711.11383v1 [stat.ML]`.

Deng, J., J. Krause, A. C. Berg, and L. Fei-Fei (2012). "Hedging Your Bets: Optimizing Accuracy-specificity Trade-offs in Large Scale Visual Recognition". In: *Computer Vision and Pattern Recognition (CVPR)*. DOI: `10.1109/CVPR.2012.6248086`.

Deng, Jia, Alexander C Berg, Kai Li, and Li Fei-Fei (2010). "What Does Classifying More Than 10,000 Image Categories Tell Us?" In: *European Conference on Computer Vision (ECCV)*. DOI: `10.1007/978-3-642-15555-0_6`.

Deng, Jia, Alexander C. Berg, and Li Fei-Fei (2011). "Hierarchical Semantic Indexing for Large Scale Image Retrieval". In: *Computer Vision and Pattern Recognition (CVPR)*. DOI: `10.1109/CVPR.2011.5995516`.

Deng, Jia, Nan Ding, Yangqing Jia, Andrea Frome, Kevin Murphy, Samy Bengio, Yuan Li, Hartmut Neven, and Hartwig Adam (2014). "Large-scale Object Classification Using Label Relation Graphs". In: *European Conference on Computer Vision (ECCV)*. DOI: `10.1007/978-3-319-10590-1_4`.

Deng, Jia, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei (2009). "ImageNet: A Large-scale Hierarchical Image Database". In: *Computer Vision and Pattern Recognition (CPVR)*. DOI: `10.1109/CVPR.2009.5206848`.

Deng, Jia, Sanjeev Satheesh, Alexander C Berg, and Fei-Fei Li (2011). "Fast and Balanced: Efficient Label Tree Learning for Large Scale Object Recognition". In: *Neural Information Processing Systems (NIPS)*.

Deselaers, Thomas and Vittorio Ferrari (2011). "Visual and Semantic Similarity in ImageNet". In: *Computer Vision and Pattern Recognition (CVPR)*. DOI: `10.1109/CVPR.2011.5995474`.

Dhall, Ankit, Anastasia Makarova, Octavian Ganea, Dario Pavllo, Michael Greeff, and Andreas Krause (2020). "Hierarchical Image Classification Using Entailment Cone Embeddings". In: *Computer Vision and Pattern Recognition Workshops (CVPR-WS)*. DOI: `10.1109/CVPRW50498.2020.00426`.

Diestel, Reinhard (2017). *Graph Theory*. Springer-Verlag Berlin Heidelberg. ISBN: 978-3-662-53621-6. DOI: `10.1007/978-3-662-53622-3`.

Dosovitskiy, Alexey, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby (2021). "An Image Is Worth 16x16 Words: Transformers for Image Recognition at Scale". In: *International Conference on Learning Representations (ICLR)*. arXiv: `2010.11929v1 [cs.CV]`.

Duan, K., D. Parikh, D. Crandall, and K. Grauman (2012). "Discovering Localized Attributes for Fine-grained Recognition". In: *Computer Vision and Pattern Recognition (CVPR)*. DOI: `10.1109/CVPR.2012.6248089`.

Everingham, Mark, Luc van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman (2010). "The Pascal Visual Object Classes (VOC) Challenge". In: *International Journal of Computer Vision* 88 (2), pp. 303–338. DOI: `10.1007/s11263-009-0275-4`.

Everingham, Mark, Luc van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman (2015). "The Pascal Visual Object Classes Challenge: A Retrospective".

In: *International Journal of Computer Vision* 111 (1), pp. 98–136. DOI: `10.1007/s11263-014-0733-5`.

Faghri, Fartash, David J. Fleet, Jamie Ryan Kiros, and Sanja Fidler (2018). “VSE++: Improving Visual-semantic Embeddings with Hard Negatives”. In: *British Machine Vision Conference (BMVC)*. arXiv: `1707.05612v4 [cs.LG]`.

Fan, J., Y. Gao, and H. Luo (2008). “Integrating Concept Ontology and Multitask Learning to Achieve More Effective Classifier Training for Multilevel Image Annotation”. In: *IEEE Transactions on Image Processing*. DOI: `10.1109/TIP.2008.916999`.

Fan, J., T. Zhao, Z. Kuang, Y. Zheng, J. Zhang, J. Yu, and J. Peng (2017). “HD-MTL: Hierarchical Deep Multi-task Learning for Large-scale Visual Recognition”. In: *IEEE Transactions on Image Processing* 26.4, pp. 1923–1938. DOI: `10.1109/TIP.2017.2667405`.

Fan, J., N. Zhou, J. Peng, and L. Gao (2015). “Hierarchical Learning of Tree Classifiers for Large-scale Plant Species Identification”. In: *IEEE Transactions on Image Processing* 24.11, pp. 4172–4184. DOI: `10.1109/TIP.2015.2457337`.

Fellbaum, Christine and George A. Miller (1990). “Folk Psychology or Semantic Entailment? Comment on Rips and Conrad (1989)”. In: *Psychological Review* 97 (4), pp. 565–570. DOI: `10.1037/0033-295X.97.4.565`.

Fergus, Rob, Hector Bernal, Yair Weiss, and Antonio Torralba (2010). “Semantic Label Sharing for Learning with Many Categories”. In: *European Conference on Computer Vision (ECCV)*. DOI: `10.1007/978-3-642-15549-9_55`.

Fitting, Melvin (2020). “Intensional Logic”. In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta. Metaphysics Research Lab, Stanford University. URL: `https://plato.stanford.edu/entries/logic-intensional`.

Freytag, Alexander, Erik Rodner, and Joachim Denzler (2014). “Selecting Influential Examples: Active Learning with Expected Model Output Changes”. In: *European Conference on Computer Vision (ECCV)*. DOI: `10.1007/978-3-319-10593-2_37`.

Freytag, Alexander, Erik Rodner, Marcel Simon, Alexander Loos, Hjalmar S Kühl, and Joachim Denzler (2016). “Chimpanzee Faces in the Wild: Log-euclidean CNNs for Predicting Identities and Attributes of Primates”. In: *German Conference on Pattern Recognition (GCPR)*. DOI: `10.1007/978-3-319-45886-1_5`.

Frome, Andrea, Greg Corrado, Jonathon Shlens, Samy Bengio, Jeffrey Dean, Marc’Aurelio Ranzato, and Tomas Mikolov (2013). “Devise: A Deep Visual-semantic Embedding Model”. In: *Neural Information Processing Systems (NIPS)*.

Fu, Ruiji, Jiang Guo, Bing Qin, Wanxiang Che, Haifeng Wang, and Ting Liu (2014). "Learning Semantic Hierarchies Via Word Embeddings". In: *Association for Computational Linguistics (ACL)*. DOI: `10.3115/v1/P14-1113`.

Ganea, Octavian, Gary Becigneul, and Thomas Hofmann (2018). "Hyperbolic Entailment Cones for Learning Hierarchical Embeddings". In: *International Conference on Machine Learning (ICML)*.

Ge, Weifeng (2018). "Deep Metric Learning with Hierarchical Triplet Loss". In: *European Conference on Computer Vision (ECCV)*. DOI: `10.1007/978-3-030-01231-1_17`.

Geirhos, Robert, Kantharaju Narayanappa, Benjamin Mitzkus, Matthias Bethge, Felix A. Wichmann, and Wieland Brendel (2020). "On the Surprising Similarities between Supervised and Self-supervised Models". In: *Neural Information Processing Systems Workshops (NeurIPS-WS)*. arXiv: `2010.08377v1 [cs.CV]`.

Gillon, Brendan S. (1990). "Ambiguity, Generality, and Indeterminacy: Tests and Definitions". In: *Synthese* 85.3, pp. 391–416. DOI: `10.1007/BF00484835`.

Giraldo-Zuluaga, Jhony-Heriberto, Augusto Salazar, Alexander Gomez Villa, and Angélica Diaz-Pulido (2017). "Recognition of Mammal Genera on Camera-trap Images Using Multi-layer Robust Principal Component Analysis and Mixture Neural Networks". In: *International Conference on Tools with Artificial Intelligence (ICTAI)*. DOI: `10.1109/ICTAI.2017.00020`.

Giraldo-Zuluaga, Jhony-Heriberto, Augusto Salazar, Alexander Gomez Villa, and Angélica Diaz-Pulido (2019). "Camera-trap Images Segmentation Using Multi-layer Robust Principal Component Analysis". In: *The Visual Computer* 35 (3), pp. 335–347. DOI: `10.1007/s00371-017-1463-9`.

Girshick, Ross (2015). "Fast R-CNN". In: *International Conference on Computer Vision (ICCV)*. DOI: `10.1109/ICCV.2015.169`.

Givant, S. (2017). *Introduction to Relation Algebras*. Springer-Verlag Cham. ISBN: 978-3-319-65234-4. DOI: `10.1007/978-3-319-65235-1`.

Glorot, Xavier and Yoshua Bengio (2010). "Understanding the Difficulty of Training Deep Feedforward Neural Networks". In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Gomez Villa, Alexander, German Diez, Augusto Salazar, and Angelica Diaz (2016). "Animal Identification in Low Quality Camera-trap Images Using Very Deep Convolutional Neural Networks and Confidence Thresholds". In: *Advances in Visual Computing*. DOI: `10.1007/978-3-319-50835-1_67`.

Gomez Villa, Alexander, Augusto Salazar, and Francisco Vargas (2017). “Towards Automatic Wild Animal Monitoring: Identification of Animal Species in Camera-trap Images Using Very Deep Convolutional Neural Networks”. In: *Ecological Informatics* 41, pp. 24–32. DOI: `10.1016/j.ecoinf.2017.07.004`.

Gonzalez, Rafael C. and Richard E. Woods (2018). *Digital Image Processing*. Pearson. ISBN: 978-0133356724.

Goo, Wonjoon, Juyong Kim, Gunhee Kim, and Sung Ju Hwang (2016). “Taxonomy-regularized Semantic Deep Convolutional Neural Networks”. In: *European Conference on Computer Vision (ECCV)*. DOI: `10.1007/978-3-319-46475-6_6`.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep Learning*. MIT Press. ISBN: 978-0262035613.

Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio (2014). “Generative Adversarial Nets”. In: *Neural Information Processing Systems (NIPS)*.

Goyal, Palash and Shalini Ghosh (2020). “Hierarchical Class-based Curriculum Loss”. In: arXiv: `2006.03629v1 [cs.LG]`.

Gray, Alasdair J. G., Carole Goble, Rafael C. Jimenez, and The Bioschemas Community (2017). “Bioschemas: From Potato Salad to Protein Annotation”. In: *International Semantic Web Conference*. URL: `http://bioschemas.org/` (visited on 02/09/2021).

Guo, Chuan, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger (2017). “On Calibration of Modern Neural Networks”. In: *International Conference on Machine Learning (ICML)*. DOI: `10.5555/3305381.3305518`.

Häfner, M., M. Liedlgruber, S. Maimone, A. Uhl, A. Vécsei, and F. Wrba (2012). “Evaluation of Cross-validation Protocols for the Classification of Endoscopic Images of Colonic Polyps”. In: *International Symposium on Computer-Based Medical Systems (CBMS)*. DOI: `10.1109/CBMS.2012.6266355`.

Harispe, Sébastien, Sylvie Ranwez, Stefan Janaqi, and Jacky Montmain (2015). *Semantic Similarity from Natural Language and Ontology Analysis*. Morgan & Claypool. DOI: `10.2200/S00639ED1V01Y201504HLT027`.

He, Guiqing, Feng Li, Qiyao Wang, Zongwen Bai, and Yuelei Xu (2021). “A Hierarchical Sampling Based Triplet Network for Fine-grained Image Classification”. In: *Pattern Recognition* 115. DOI: `10.1016/j.patcog.2021.107889`.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2015). “Delving Deep into Rectifiers: Surpassing Human-level Performance on Imagenet Classification”. In: *International Conference on Computer Vision (ICCV)*. DOI: `10.1109/ICCV.2015.123`.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2016a). "Deep Residual Learning for Image Recognition". In: *Computer Vision and Pattern Recognition (CVPR)*. DOI: `10.1109/CVPR.2016.90`.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2016b). "Identity Mappings in Deep Residual Networks". In: *European Conference on Computer Vision (ECCV)*. DOI: `10.1007/978-3-319-46493-0_38`.

Hwang, Sung Ju (2013). "Discriminative Object Categorization with External Semantic Knowledge". PhD thesis. The University of Texas at Austin.

Hwang, Sung Ju and Leonid Sigal (2014). "A Unified Semantic Embedding: Relating Taxonomies and Attributes". In: *Neural Information Processing Systems Workshops (NIPS-WS)*. arXiv: `1411.5879v2 [cs.CV]`.

Ioffe, Sergey and Christian Szegedy (2015). "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *International Conference on Machine Learning (ICML)*. arXiv: `1502.03167v3 [cs.LG]`.

Jayathilaka, Mirantha, Tingting Mu, and Uli Sattler (2020). "Visual-semantic Embedding Model Informed by Structured Knowledge". In: *European Starting AI Researchers' Symposium (STAIRS)*. arXiv: `2009.10026v1 [cs.CV]`.

Jia, Menglin, Mengyun Shi, Mikhail Sirotenko, Yin Cui, Bharath Hariharan, Claire Cardie, and Serge Belongie (2019). *The Fashionpedia Ontology and Fashion Segmentation Dataset*. Tech. rep. Cornell University.

Jia, Yangqing, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell (2014). "Caffe: Convolutional Architecture for Fast Feature Embedding". In: *International Conference on Multimedia (MM)*. DOI: `10.1145/2647868.2654889`.

Jiang, Jay J. and David W. Conrath (1997). "Semantic Similarity Based on Corpus Statistics and Lexical Taxonomy". In: *Computational Linguistics and Speech Processing (ROCLING)*. arXiv: `9709008 [cmp-lg]`.

Jo, J., J. Kung, S. Lee, and Y. Lee (2019). "Similarity-based Lstm Architecture for Energy-efficient Edge-level Speech Recognition". In: *International Symposium on Low Power Electronics and Design (ISLPED)*. DOI: `10.1109/ISLPED.2019.8824862`.

Käding, Christoph, Erik Rodner, Alexander Freytag, and Joachim Denzler (2016a). "Active and Continuous Exploration with Deep Neural Networks and Expected Model Output Changes". In: *Neural Information Processing Systems Workshops (NIPS-WS)*. arXiv: `1612.06129 [cs.CV]`.

Käding, Christoph, Erik Rodner, Alexander Freytag, and Joachim Denzler (2016b). “Fine-tuning Deep Neural Networks in Continuous Learning Scenarios”. In: *Asian Conference on Computer Vision Workshops (ACCV-WS)*. DOI: `10.1007/978-3-319-54526-4_43`.

Käding, Christoph, Erik Rodner, Alexander Freytag, Oliver Mothes, Björn Barz, and Joachim Denzler (2018). “Active Learning for Regression Tasks with Expected Model Output Changes”. In: *British Machine Vision Conference (BMVC)*.

Karthik, Shyamgopal, Ameya Prabhu, Puneet K. Dokania, and Vineet Gandhi (2021). “No Cost Likelihood Manipulation at Test Time for Making Better Mistakes in Deep Networks”. In: *International Conference on Learning Representations (ICLR)*. arXiv: `2104.00795v1 [cs.LG]`.

Kingma, Diederik P. and Jimmy Ba (2014). “Adam: A Method for Stochastic Optimization”. In: arXiv: `1412.6980v9 [cs.LG]`.

Kiritchenko, Svetlana, Stan Matwin, and A. Fazel Famili (2005). “Functional Annotation of Genes Using Hierarchical Text Categorization”. In: *Association for Computational Linguistics Workshops (ACL-WS)*.

Kirkpatrick, James, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell (2017). “Overcoming Catastrophic Forgetting in Neural Networks”. In: *Proceedings of the National Academy of Sciences (PNAS)* 114.13, pp. 3521–3526. DOI: `10.1073/pnas.1611835114`.

Kohli, Harsh (2021). “Transfer Learning and Augmentation for Word Sense Disambiguation”. In: *European Conference on Information Retrieval (ECIR)*.

Krause, Jonathan, Michael Stark, Jia Deng, and Li Fei-Fei (2013). “3D Object Representations for Fine-grained Categorization”. In: *International Conference on Computer Vision Workshops (ICCV-WS)*. DOI: `10.1109/ICCVW.2013.77`.

Krishna, Ranjay, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A. Shamma, Michael S. Bernstein, and Li Fei-Fei (2017). “Visual Genome: Connecting Language and Vision Using Crowdsourced Dense Image Annotations”. In: *International Journal of Computer Vision (IJCV)* 123.1, pp. 32–73. DOI: `10.1007/s11263-016-0981-7`.

Krizhevsky, Alex and Geoffrey Hinton (2009). *Learning Multiple Layers of Features from Tiny Images*. Tech. rep. University of Toronto.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton (2012). "ImageNet Classification with Deep Convolutional Neural Networks". In: *Neural Information Processing Systems (NIPS)*. DOI: `10.5555/2999134.2999257`.

Kühl, Hjalmar (2008). *Best Practice Guidelines for the Surveys and Monitoring of Great Ape Populations*. International Union for Conservation of Nature and Natural Resource. ISBN: 978-2-8317-1062-4.

Kühnl, T., F. Kummert, and J. Fritsch (2012). "Spatial Ray Features for Real-time Ego-lane Extraction". In: *Intelligent Transportation Systems (ITS)*. DOI: `10.1109/ITSC.2012.6338740`.

La Grassa, Riccardo, Ignazio Gallo, and Nicola Landro (2021). "Learn Class Hierarchy Using Convolutional Neural Networks". In: *Applied Intelligence*. DOI: `10.1007/s10489-020-02103-6`.

Lawry, Jonathan (2001). "Label Semantics: A Formal Framework for Modeling with Words". In: *European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU)*. DOI: `10.1007/3-540-44652-4_33`.

LeCun, Y., L. Bottou, Y. Bengio, and P. Haffner (1988). "Gradient-based Learning Applied to Document Recognition". In: *Proceedings of the IEEE*. 11. DOI: `10.1109/5.726791`.

Li, Xiaoni, Yucan Zhou, Yu Zhou, and Weiping Wang (2021). "MMF: Multi-task Multi-structure Fusion for Hierarchical Image Classification". In: arXiv: `2107.00808 [cs.CV]`.

Lin, Tsung-Yi, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick (2014). "Microsoft COCO: Common Objects in Context". In: *European Conference on Computer Vision (ECCV)*. DOI: `10.1007/978-3-319-10602-1_48`.

Lin, Wanyu, Zhaolin Gao, and Baochun Li (2020). "Shoestring: Graph-based Semi-supervised Classification with Severely Limited Labeled Data". In: *Computer Vision and Pattern Recognition (CVPR)*. DOI: `10.1109/CVPR42600.2020.00423`.

Linnæus, Carl (1758). *Systema naturæ per regna tria naturæ, secundum classes, ordines, genera, species, cum characteribus, differentiis, synonymis, locis*. 10th ed. Vol. 1. Holmiæ. URL: `http://www.biodiversitylibrary.org/item/10277` (visited on 02/11/2021).

Liu, Wei, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg (2016). "SSD: Single Shot Multibox Detector". In: *European Conference on Computer Vision (ECCV)*. DOI: `10.1007/978-3-319-46448-0_2`.

Loper, Edward and Steven Bird (2002). “NLTK: The Natural Language Toolkit”. In: *Association for Computational Linguistics (ACL)*. DOI: `10.3115/1118108.1118117`.

Lorhardus, Jacobus (1606). *Theatrum Philisophicum. Ogdoas Scholastica, continens Diagraphen Typicam artium: Grammatices (Latinae, Graecae), Logices, Rhetorices, Astronomices, Ethices, Physices, Metaphysices, seu Ontologiae*. 1st ed. Sangalli.

Loshchilov, Ilya and Frank Hutter (2017). “SGDR: Stochastic Gradient Descent with Warm Restarts”. In: *International Conference on Learning Representations (ICLR)*. arXiv: `1608.03983 [cs.LG]`.

Lu, Yijun (1997). “Concept Hierarchy in Data Mining. Specification, Generation and Implementation”. MA thesis. Simon Fraser University.

Maedche, Alexander and Steffen Staab (2001). *Comparing Ontologies - Similarity Measures and a Comparison Study*. Tech. rep. Institute AIFB, University of Karlsruhe.

Magnini, Bernardo, Carlo Strapparava, Giovanni Pezzulo, and Alfio Gliozzo (2001). “Using Domain Information for Word Sense Disambiguation”. In: *Evaluating Word Sense Disambiguation Systems (SENSEVAL)*.

Mahajan, Dhruv, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten (2008). “Exploring the Limits of Weakly Supervised Pretraining”. In: *European Conference on Computer Vision (ECCV)*. DOI: `10.1007/978-3-030-01216-8_12`.

Manning, Christopher D., Prabhakar Raghavan, and Hinrich Schütze (2008). *Introduction to Information Retrieval*. Cambridge University Press. ISBN: 9780521865715.

Marszalek, Marcin and Cordelia Schmid (2007). “Semantic Hierarchies for Visual Object Recognition”. In: *Computer Vision and Pattern Recognition (CVPR)*. DOI: `10.1109/CVPR.2007.383272`.

Masters, Dominic and Carlo Luschi (2018). “Revisiting Small Batch Training for Deep Neural Networks”. In: arXiv: `1804.07612v1 [cs.LG]`.

McAuley, Julian J, Arnau Ramisa, and Tibério S Caetano (2013). “Optimization of Robust Loss Functions for Weakly-labeled Image Taxonomies”. In: *International Journal of Computer Vision (IJCV)* 104.3, pp. 343–361. DOI: `10.1007/s11263-012-0561-4`.

McCulloch, Warren S. and Walter Pitts (1943). “A Logical Calculus of the Ideas Immanent in Nervous Activity”. In: *The bulletin of mathematical biophysics* 5 (4), pp. 115–133. DOI: `10.1007/BF02478259`.

Milbich, Timo, Omair Ghori, Ferran Diego, and Björn Ommer (2020). “Unsupervised Representation Learning by Discovering Reliable Image Relations”. In: *Pattern Recognition* 102. DOI: `10.1016/j.patcog.2019.107107`.

Miller, George A. (1995). “Wordnet: A Lexical Database for English”. In: *Communications of the ACM* 38 (11), pp. 39–41. DOI: `10.1145/219717.219748`.

Mo, Kaichun, Shilin Zhu, Angel X Chang, Li Yi, Subarna Tripathi, Leonidas J Guibas, and Hao Su (2019). “PartNet: A Large-scale Benchmark for Fine-grained and Hierarchical Part-level 3d Object Understanding”. In: *Computer Vision and Pattern Recognition (CVPR)*. DOI: `10.1109/CVPR.2019.00100`.

Nair, Vinod and Geoffrey E. Hinton (2010). “Rectified Linear Units Improve Restricted Boltzmann Machines”. In: *International Conference on Machine Learning (ICML)*. DOI: `10.5555/3104322.3104425`.

Narayana, Pradyumna, Aniket Pednekar, Abishek Krishnamoorthy, Kazoo Sone, and Sugato Basu (2019). “HUSE: Hierarchical Universal Semantic Embeddings”. In: arXiv: `1911.05978v1 [cs.CV]`.

Navigli, Roberto and Simone Paolo Ponzetto (2012). “BabelNet: The Automatic Construction, Evaluation and Application of a Wide-coverage Multilingual Semantic Network.” In: *Artifical Intelligence* 193, pp. 217–250. DOI: `10.1016/j.artint.2012.07.001`.

Neches, Robert, Richard E. Fikes, Tim Finin, Thomas Gruber, Ramesh Patil, Ted Senator, and William R. Swartout (1991). “Enabling Technology for Knowledge Sharing”. In: *AI Magazine* 12.3. DOI: `10.1609/aimag.v12i3.902`.

Nickel, Maximilian and Douwe Kiela (2017). “Poincaré Embeddings for Learning Hierarchical Representations”. In: *Neural Information Processing Systems (NIPS)*. arXiv: `1705.08039v2 [cs.AI]`.

Niemann, Heinrich (1983). *Klassifikation von Mustern*. Springer-Verlag Berlin Heidelberg. ISBN: 978-3-540-12642-3. DOI: `10.1007/978-3-642-47517-7`.

Norouzi, Mohammad, Tomas Mikolov, Samy Bengio, Yoram Singer, Jonathon Shlens, Andrea Frome, Greg Corrado, and Jeffrey Dean (2014). “Zero-shot Learning by Convex Combination of Semantic Embeddings”. In: *International Conference on Learning Representations (ICLR)*. arXiv: `1312.5650v3 [cs.LG]`.

Norouzzadeh, Mohammed Sadegh, Anh Nguyen, Margaret Kosmala, Ali Swanson, Meredith Palmer, Craig Packer, and Jeff Clune (2017). “Automatically Identifying, Counting, and Describing Wild Animals in Camera-trap Images with Deep Learning”. In: arXiv: `1703.05830v5 [cs.CV]`.

Oliva, Aude and Antonio Torralba (2001). “Modeling the Shape of the Scene: A Holistic Representation of the Spatial Envelope”. In: *International Journal of Computer Vision (IJCV)* 42 (3), pp. 145–175. DOI: `10.1023/A:1011139631724`.

Oxford English Dictionary (2020). *Learn, V.* Oxford University Press. URL: `https://www.oed.com/view/Entry/106716?rskey=7c9MaO&amp;result=1&amp;isAdvanced=false` (visited on 10/07/2020).

Papadopoulos, Dim P., Jasper R. R. Uijlings, Frank Keller, and Vittorio Ferrari (June 2016). “We Don’t Need No Bounding-boxes: Training Object Class Detectors Using Only Human Verification”. In: *Computer Vision and Pattern Recognition (CVPR)*. DOI: `10.1109/CVPR.2016.99`.

Parkhi, Omkar M., Andrea Vedaldi, and Andrew Zisserman (2015). “Deep Face Recognition”. In: *British Machine Vision Conference (BMVC)*. DOI: `10.5244/C.29.41`.

Parnell, Richard J. (2002). “Group Size and Structure in Western Lowland Gorillas (*Gorilla gorilla gorilla*) at Mbeli Bai, Republic of Congo”. In: *American Journal of Primatology* 56 (4), pp. 193–206. DOI: `10.1002/ajp.1074`.

Pease, Adam, Christiane Fellbaum, and Piek Vossen (2008). “Building the Global WordNet Grid”. In: *International Congress of Linguists (CIL18)*.

Penzel, Niklas (2018). “Lebenslanges Lernen Von Klassifikationssystemen Ohne Vorwissen Und Mit Intelligenter Datenhaltung”. MA thesis. Friedrich Schiller University Jena.

Pianta, Emanuele, Luisa Bentivogli, and Christian Girardi (2002). “MultiWordNet: Developing an Aligned Multilingual Database”. In: *First International Conference on Global WordNet*.

Proença, Hugo, Ehsan Yaghoubi, and Pendar Alirezazadeh (2020). “A Quadruplet Loss for Enforcing Semantically Coherent Embeddings in Multi-output Classification Problems”. In: *IEEE Transactions on Information Forensics and Security* 16, pp. 800–811. DOI: `10.1109/TIFS.2020.3023304`.

Qian, Qi, Lei Shang, Baigui Sun, Juhua Hu, Hao Li, and Rong Jin (2019). “SoftTriple Loss: Deep Metric Learning without Triplet Sampling”. In: *International Conference on Computer Vision (ICCV)*. DOI: `10.1109/ICCV.2019.00655`.

Qin, Zengchang and Jonathan Lawry (2004). “A Tree Structured Classification Model Based on Label Semantics”. In: *Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU)*.

Redmon, Joseph, Santosh Divvala, Ross Girshick, and Ali Farhadi (2016). "You Only Look Once: Unified, Real-time Object Detection". In: *Computer Vision and Pattern Recognition (CVPR)*. DOI: `10.1109/CVPR.2016.91`.

Resnik, Philip (1995). "Using Information Content to Evaluate Semantic Similarity in a Taxonomy". In: *International Joint Conference on Artificial Intelligence (IJCAI)*. DOI: `10.5555/1625855.1625914`.

Rosenblatt, Frank (1958). "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain." In: *Psychological Review* 65 (6), pp. 386–408. DOI: `10.1037/h0042519`.

Roy, Deboleena, Priyadarshini Panda, and Kaushik Roy (2020). "Tree-CNN: A Hierarchical Deep Convolutional Neural Network for Incremental Learning". In: *Neural Networks* 121, pp. 148–160. DOI: `10.1016/j.neunet.2019.09.010`.

Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams (1986). "Learning Representations by Back-propagating Errors". In: *Nature* 323 (6088), pp. 533–536. DOI: `10.1038/323533a0`.

Russakovsky, Olga, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei (2015). "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision (IJCV)* 115 (3), pp. 211–252. DOI: `10.1007/s11263-015-0816-y`.

Samplawski, Colin, Heesung Kwon, Erik Learned-Miller, and Benjamin M. Marlin (2019). "Integrating Propositional and Relational Label Side Information for Hierarchical Zero-shot Image Classification". In: arXiv: `1902.05492v1 [cs.CV]`.

Sanakoyeu, Artsiom, Vadim Tschernezki, Uta Buchler, and Björn Ommer (2019). "Divide and Conquer the Embedding Space for Metric Learning". In: *Computer Vision and Pattern Recognition (CVPR)*. DOI: `10.1109/CVPR.2019.00056`.

Sánchez, David, Montserrat Batet, David Isern, and Aida Valls (2012). "Ontology-based Semantic Similarity: A New Feature-based Approach". In: 39.9, pp. 7718–7728. DOI: `10.1016/j.eswa.2012.01.082`.

Schölkopf, Bernhard, Robert C Williamson, Alexander J Smola, John Shawe-Taylor, and John C Platt (1999). "Support Vector Method for Novelty Detection". In: *Neural Information Processing Systems (NIPS)*. DOI: `10.5555/3009657.3009740`.

Schroff, Florian, Dmitry Kalenichenko, and James Philbin (2015). "Facenet: A Unified Embedding for Face Recognition and Clustering". In: *Computer Vision and Pattern Recognition (CVPR)*. DOI: `10.1109/CVPR.2015.7298682`.

Seeland, Marco, Michael Rzanny, David Boho, Jana Wäldchen, and Patrick Mäder (2019). "Image-based Classification of Plant Genus and Family for Trained and Untrained Plant Species". In: *BMC Bioinformatics* 20.4 (1), pp. 1–13. DOI: `10.1186/s12859-018-2474-x`.

Settles, Burr (2009). *Active Learning Literature Survey*. Tech. rep. University of Wisconsin-Madison.

Shalev-Shwartz, Shai and Shai Ben-David (2014). *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press. ISBN: 978-1-107-05713-5. DOI: `10.5555/2621980`.

Sharma, Piyush, Nan Ding, Sebastian Goodman, and Radu Soricut (2018). "Conceptual Captions: A Cleaned, Hypernymed, Image Alt-text Dataset for Automatic Image Captioning". In: *Association for Computational Linguistics (ACL)*. DOI: `10.18653/v1/P18-1238`.

Sidorov, Oleksii, Ronghang Hu, Marcus Rohrbach, and Amanpreet Singh (2020). "TextCaps: A Dataset for Image Captioning with Reading Comprehension". In: *European Conference on Computer Vision (ECCV)*. DOI: `10.1007/978-3-030-58536-5_44`.

Silla, Carlos and Alex Freitas (2011). "A Survey of Hierarchical Classification across Different Application Domains". In: *Data Mining and Knowledge Discrovery* 22 (1), pp. 31–72. DOI: `10.1007/s10618-010-0175-9`.

Sirakov, Nikolay, Abdullah N. Arslan, Christian F. Hempelmann, Salvatore Attardo, and Grady P. Blount (2015). "Firearms Identification through Partonomy". In: *SPIE Newsroom*. DOI: `10.1117/2.1201507.005849`.

Song, Hyun Oh, Ross Girshick, Stefanie Jegelka, Julien Mairal, Zaid Harchaoui, and Trevor Darrell (2014). "On Learning to Localize Objects with Minimal Supervision". In: *International Conference on Machine Learning (ICML)*. DOI: `10.5555/3044805.3045072`.

Srivastava, Nitish and Ruslan Salakhutdinov (2013). "Discriminative Transfer Learning with Tree-based Priors". In: *Neural Information Processing Systems (NIPS)*.

Stretcu, Otilia, Emmanouil Antonios Platanios, Tom M. Mitchell, and Barnabás Póczos (2021). "Coarse-to-fine Curriculum Learning". In: arXiv: `2106.04072 [cs.AI]`.

Strubell, Emma, Ananya Ganesh, and Andrew McCallum (2019). "Energy and Policy Considerations for Deep Learning in Nlp". In: *Association for Computational Linguistics (ACL)*. DOI: `10.18653/v1/P19-1355`.

Sun, Chen, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta (2017a). "Revisiting Unreasonable Effectiveness of Data in Deep Learning Era". In: *International Conference on Computer Vision (ICCV)*. DOI: `10.1109/ICCV.2017.97`.

Sun, Xu, Bingzhen Wei, Xuancheng Ren, and Shuming Ma (2017b). "Label Embedding Network: Learning Label Representation for Soft Training of Deep Networks". In: arXiv: `1710.10393v1 [cs.LG]`.

Swanson, Alexandra, Margaret Kosmala, Chris Lintott, Robert Simpson, Arfon Smith, and Craig Packer (2015). "Snapshot Serengeti, High-frequency Annotated Camera Trap Images of 40 Mammalian Species in an African Savanna". In: *Scientific Data* 2 (1). DOI: `10.1038/sdata.2015.26`.

Tan, Mingxing and Quoc Le (2019). "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks". In: *International Conference on Machine Learning (ICML)*.

Taskar, Ben, Carlos Guestrin, and Daphne Koller (2004). "Max-margin Markov Networks". In: *Neural Information Processing Systems (NIPS)*.

Thomas, Chris D, Alison Cameron, Rhys E Green, Michel Bakkenes, Linda J Beaumont, Yvonne C Collingham, Barend FN Erasmus, Marinez Ferreira De Siqueira, Alan Grainger, and Lee Hannah (2004). "Extinction Risk from Climate Change". In: *Nature* 427.6970, pp. 145–148. DOI: `10.1038/nature02121`.

Tilman, David, Robert M. May, Clarence L Lehman, and Martin A. Nowak (1994). "Habitat Destruction and the Extinction Debt". In: *NAture* 371.6492, pp. 65–66. DOI: `10.1038/371065a0`.

Tonioni, Alessio and Luigi Di Stefano (2019). "Domain Invariant Hierarchical Embedding for Grocery Products Recognition". In: *Computer Vision and Image Understanding (CVIU)* 182, pp. 81–92. DOI: `10.1016/j.cviu.2019.03.005`.

Torralba, Antonio, Rob Fergus, and William T. Freeman (2008). "80 Million Tiny Images: A Large Data Set for Nonparametric Object and Scene Recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30.11, pp. 1958–1970. DOI: `10.1109/TPAMI.2008.128`.

Valiant, Leslie G (1984). "A Theory of the Learnable". In: *Communications of the ACM* 27.11, pp. 1134–1142. DOI: `10.1145/1968.1972`.

Van Horn, G., O. Mac Aodha, Y. Song, Y. Cui, C. Sun, A. Shepard, H. Adam, P. Perona, and S. Belongie (2018). "The iNaturalist Species Classification and Detection Dataset". In: *Computer Vision and Pattern Recognition (CVPR)*. DOI: `10.1109/CVPR.2018.00914`.

Van Horn, G., S. Branson, R. Farrell, S. Haber, J. Barry, P. Ipeirotis, P. Perona, and S. Belongie (2015). "Building a Bird Recognition App and Large Scale Dataset with Citizen Scientists: The Fine Print in Fine-grained Dataset Collection". In: *Computer Vision and Pattern Recognition (CVPR)*. DOI: `10.1109/CVPR.2015.7298658`.

Vapnik, Vladimir Naumovich (1998). *Statistical Learning Theory*. Wiley-Interscience. ISBN: 978-0-471-03003-4.

Vapnik, Vladimir Naumovich and Alexey Yakovlevich Chervonenkis (1971). "On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities". In: *Theory of Probability & Its Applications* 16.2, pp. 264–280. DOI: `10.1137/1116025`.

Veit, Andreas, Tomas Matera, Lukas Neumann, Jiri Matas, and Serge Belongie (2016). "COCO-Text: Dataset and Benchmark for Text Detection and Recognition in Natural Images". In: arXiv: `1601.07140v2 [cs.CV]`.

Vendrov, Ivan, Ryan Kiros, Sanja Fidler, and Raquel Urtasun (2016). "Order-Embeddings of Images and Language". In: *International Conference on Learning Representations (ICLR)*. arXiv: `1511.06361 [cs.LG]`.

Verma, N., D. Mahajan, S. Sellamanickam, and V. Nair (2012a). "Learning Hierarchical Similarity Metrics". In: *Computer Vision and Pattern Recognition (CPVR)*. DOI: `10.1109/CVPR.2012.6247938`.

Verma, Nakul, Dhruv Mahajan, Sundararajan Sellamanickam, and Vinod Nair (2012b). "Learning Hierarchical Similarity Metrics". In: *Computer Vision and Pattern Recognition (CVPR)*. DOI: `10.1109/CVPR.2012.6247938`.

Vié, Jean-Christophe, Craig Hilton-Taylor, and Simon N. Stuart, eds. (2009). *Wildlife in a Changing World: An Analysis of the 2008 Iucn Red List of Threatened Species*. International Union for Conservation of Nature and Natural Resource. ISBN: 978-84-96553-63-7.

Vogel, Gretchen (May 2017). "Where Have All the Insects Gone?" In: *Science*. DOI: `10.1126/science.aal1160`.

W3C OWL Working Group, ed. (Dec. 11, 2012). *OWL 2 Web Ontology Language*. URL: `https://www.w3.org/TR/owl2-overview/` (visited on 02/09/2021).

Wang, Jian, Feng Zhou, Shilei Wen, Xiao Liu, and Yuanqing Lin (2017a). "Deep Metric Learning with Angular Loss". In: *International Conference on Computer Vision (ICCV)*. DOI: `10.1109/ICCV.2017.283`.

Wang, Keze, Dongyu Zhang, Ya Li, Ruimao Zhang, and Liang Lin (2017b). "Cost-effective Active Learning for Deep Image Classification". In: *IEEE Transactions on*

Circuits and Systems for Video Technology 27.12, pp. 2591–2600. DOI: `10.1109/TCSVT.2016.2589879`.

Wang, Xiaofei, Yiwen Han, Victor CM Leung, Dusit Niyato, Xueqiang Yan, and Xu Chen (2020). “Convergence of Edge Computing and Deep Learning: A Comprehensive Survey”. In: *IEEE Communications Surveys & Tutorials* 22.2, pp. 869–904. DOI: `10.1109/COMST.2020.2970550`.

Wang, Yu, Zhou Wang, Qinghua Hu, Yucan Zhou, and Honglei Su (2021). “Hierarchical Semantic Risk Minimization for Large-scale Classification”. In: *IEEE Transactions on Cybernetics*, pp. 1–13. DOI: `10.1109/TCYB.2021.3059631`.

Wang, Yu Emma, Gu-Yeon Wei, and David Brooks (2019). “Benchmarking TPU, GPU, and CPU Platforms for Deep Learning”. In: arXiv: `1907.10701v4 [cs.LG]`.

Wang, Zhou, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli (2004). “Image Quality Assessment: From Error Visibility to Structural Similarity”. In: *IEEE Transactions on Image Processing (TIP)* 13.4, pp. 600–612. DOI: `10.1109/TIP.2003.819861`.

Washburne, J. N. (1936). “The Definition of Learning”. In: *Journal of Educational Psychology* 27 (8), pp. 603–611. DOI: `10.1037/h0060154`.

Wei, Longhui, Lingxi Xie, Jianzhong He, Jianlong Chang, Xiaopeng Zhang, Wengang Zhou, Houqiang Li, and Qi Tian (2020). “Can Semantic Labels Assist Self-supervised Visual Representation Learning?” In: arXiv: `2011.08621v1 [cs.CV]`.

Welinder, P., S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona (2010). *Caltech-UCSD Birds 200*. Tech. rep. CNS-TR-2010-001. California Institute of Technology. URL: `http://www.vision.caltech.edu/visipedia/CUB-200.html` (visited on 05/10/2021).

Wille, Rudolf (1992). “Concept Lattices and Conceptual Knowledge Systems”. In: *Computers & Mathematics with Applications* 23.6, pp. 493–515. DOI: `10.1016/0898-1221(92)90120-7`.

Wille, Rudolf (2005). “Formal Concept Analysis As Mathematical Theory of Concepts and Concept Hierarchies”. In: *Formal Concept Analysis. Foundations and Applications*. Ed. by Bernhard Ganter, Gerd Stumme, and Rudolf Wille. Springer-Verlag Berlin Heidelberg, pp. 1–33. ISBN: 978-3-540-27891-7. DOI: `10.1007/11528784_1`.

Wu, Tz-Ying, Pedro Morgado, Pei Wang, Chih-Hui Ho, and Nuno Vasconcelos (2020). “Solving Long-tailed Recognition with Deep Realistic Taxonomic Classifier”. In: *European Conference on Computer Vision (ECCV)*. DOI: `10.1007/978-3-030-58598-3_11`.

Xie, S., T. Yang, Xiaoyu Wang, and Yuanqing Lin (2015). “Hyper-class Augmented and Regularized Deep Learning for Fine-grained Image Classification”. In: *Computer Vision and Pattern Recognition (CVPR)*. DOI: `10.1109/CVPR.2015.7298880`.

Yan, Zhicheng, Hao Zhang, Robinson Piramuthu, Vignesh Jagadeesh, Dennis DeCoste, Wei Di, and Yizhou Yu (2015). “HD-CNN: Hierarchical Deep Convolutional Neural Networks for Large Scale Visual Recognition”. In: *International Conference on Computer Vision (ICCV)*. DOI: `10.1109/ICCV.2015.314`.

Yang, Jingkang, Weirong Chen, Litong Feng, Xiaopeng Yan, Huabin Zheng, and Wayne Zhang (2020). “Webly Supervised Image Classification with Metadata: Automatic Noisy Label Correction Via Visual-semantic Graph”. In: *ACM International Conference on Multimedia (MM)*. DOI: `10.1145/3394171.3413952`.

Yerushalmy, Jacob (1947). “Statistical Problems in Assessing Methods of Medical Diagnosis, with Special Reference to X-ray Techniques”. In: *Public Health Reports* 62.40, pp. 1432–1449. DOI: `10.2307/4586294`.

Younes, Zoulficar, Fahed Abdallah, and Thierry Denœux (2010). “Evidential Multi--label Classification Approach to Learning from Data with Imprecise Labels”. In: *International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU)*. DOI: `10.1007/978-3-642-14049-5_13`.

Zhai, Xiaohua, Avital Oliver, Alexander Kolesnikov, and Lucas Beyer (2019). “S4L: Self-supervised Semi-supervised Learning”. In: *International Conference on Computer Vision (ICCV)*. DOI: `10.1109/ICCV.2019.00156`.

Zhang, Jingqing, Piyawat Lertvittayakumjorn, and Yike Guo (2019). “Integrating Semantic Knowledge to Tackle Zero-shot Text Classification”. In: *Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*. arXiv: `1903.12626 [cs.CL]`.

Zhang, Ruisi, Luntian Mou, and Pengtao Xie (2020). “TreeGAN: Incorporating Class Hierarchy into Image Generation”. In: arXiv: `2009.07734v1 [cs.CV]`.

Zhang, Xiaofan, Feng Zhou, Yuanqing Lin, and Shaoting Zhang (2016). “Embedding Label Structures for Fine-grained Feature Representation”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. DOI: `10.1109/CVPR.2016.126`.

Zhao, T., B. Zhang, M. He, W. Zhang, N. Zhou, J. Yu, and J. Fan (2018). “Embedding Visual Hierarchy with Deep Networks for Large-scale Visual Recognition”. In: *IEEE Transactions on Image Processing* 27.10, pp. 4740–4755. DOI: `10.1109/TIP.2018.2845118`.

Zhou and Chellappa (1988). “Computation of Optical Flow Using a Neural Network”. In: *International Conference on Neural Networks (ICNN)*. DOI: `10.1109/ICNN.1988.23914`.

Zhou, Zhi-Hua (2017). “A Brief Introduction to Weakly Supervised Learning”. In: *National Science Review* 5.1, pp. 44–53. DOI: `10.1093/nsr/nwx106`.

Zhou, Zili, Yanna Wang, and Junzhong Gu (2008). “A New Model of Information Content for Semantic Similarity in WordNet”. In: *Future Generation Communication and Networking (FGNCS)*. DOI: `10.1109/FGCNS.2008.16`.

Zhu, Xinqi and Michael Bain (2017). “B-CNN: Branch Convolutional Neural Network for Hierarchical Classification”. In: arXiv: `1709.09890v2 [cs.CV]`.

List of Own Publications

Journal Articles

Theiß, Christoph, **Clemens-Alexander Brust**, and Joachim Denzler (2018). “Dataless Black-box Model Comparison”. In: *Pattern Recognition and Image Analysis (PRIA)* 28 (4), pp. 676–683. DOI: 10.1134/S1054661818040272.

Brust, Clemens-Alexander, Christoph Käding, and Joachim Denzler (2020). “Active and Incremental Learning with Weak Supervision”. In: *Künstliche Intelligenz* 34 (2), pp. 165–180. DOI: 10.1007/s13218-020-00631-4.

Bodesheim, Paul, Jan Blunk, Matthias Körschens, **Clemens-Alexander Brust**, Christoph Käding, and Joachim Denzler (2022). “Pre-trained Models Are Not Enough: Active and Lifelong Learning Is Important for Long-term Visual Monitoring of Mammals in Biodiversity Research. Individual identification and attribute prediction with image features from deep neural networks and decoupled decision models applied to elephants and great apes”. In: *Mammalian Biology*. Accepted for publication.

Conference Publications

Brust, Clemens-Alexander, Sven Sickert, Marcel Simon, Erik Rodner, and Joachim Denzler (2015a). “Convolutional Patch Networks with Spatial Prior for Road Detection and Urban Scene Understanding”. In: *Computer Vision Theory and Applications (VISAPP)*. DOI: 10.5220/0005355105100517.

Denzler, Joachim, Christoph Käding, and **Clemens-Alexander Brust** (2018). “Keeping the Human in the Loop: Towards Automatic Visual Monitoring in Biodiversity Research”. In: *International Conference on Ecological Informatics (ICEI)*.

Arlt, Marie, Jack Peter, Sven Sickert, **Clemens-Alexander Brust**, Joachim Denzler, and Andreas Stallmach (2019). “Automated Polyp Differentiation on Coloscopic Data Using Semantic Segmentation with CNNs”. In: *Endoscopy* 51 (4). DOI: 10.1055/s-0039-1681180.

Brust, Clemens-Alexander and Joachim Denzler (2019a). “Integrating Domain Knowledge: Using Hierarchies to Improve Deep Classifiers”. In: *Asian Conference on Pattern Recognition (ACPR)*. DOI: 10.1007/978-3-030-41404-7_1.

Brust, Clemens-Alexander and Joachim Denzler (2019b). "Not Just a Matter of Semantics: The Relationship between Visual Similarity and Semantic Similarity". In: *German Conference on Pattern Recognition (GCPR)*. DOI: `10.1007/978-3-030-33676-9_29`.

Brust, Clemens-Alexander, Christoph Käding, and Joachim Denzler (2019). "Active Learning for Deep Object Detection". In: *Computer Vision Theory and Applications (VISAPP)*. DOI: `10.5220/0007248601810190`.

Hoffmann, Stefan, **Clemens-Alexander Brust**, Maha Shadaydeh, and Joachim Denzler (2019). "Registration of High Resolution Sar and Optical Satellite Imagery Using Fully Convolutional Networks". In: *International Geoscience and Remote Sensing Symposium (IGARSS)*. DOI: `10.1109/IGARSS.2019.8898714`.

Brust, Clemens-Alexander, Björn Barz, and Joachim Denzler (2021b). "Making Every Label Count: Handling Semantic Imprecision by Integrating Domain Knowledge". In: *International Conference on Pattern Recognition (ICPR) 2020*. DOI: `10.1109/ICPR48806.2021.9413283`.

Penzel, Niklas, Christian Reimers, **Clemens-Alexander Brust**, and Joachim Denzler (2021). "Investigating the Consistency of Uncertainty Sampling in Deep Active Learning". In: *German Conference on Pattern Recognition (DAGM GCPR)*. DOI: `10.1007/978-3-030-92659-5_10`.

Brust, Clemens-Alexander, Björn Barz, and Joachim Denzler (2022). "Self-Supervised Learning from Semantically Imprecise Data". In: *Computer Vision Theory and Applications (VISAPP)*.

Workshop Publications

Brust, Clemens-Alexander, Sven Sickert, Marcel Simon, Erik Rodner, and Joachim Denzler (2015b). "Efficient Convolutional Patch Networks for Scene Understanding". In: *Computer Vision and Pattern Recognition Workshops (CVPR-WS)*.

Brust, Clemens-Alexander, Tilo Burghardt, Milou Groenenberg, Christoph Käding, Hjalmar S. Kühl, Marie L. Manguette, and Joachim Denzler (2017). "Towards Automated Visual Monitoring of Individual Gorillas in the Wild". In: *International Conference on Computer Vision Workshops (ICCV-WS)*. DOI: `10.1109/ICCVW.2017.333`.

Brust, Clemens-Alexander, Christoph Käding, and Joachim Denzler (2017). "You Have to Look More Than Once: Active and Continuous Exploration Using Yolo". In: *Computer Vision and Pattern Recognition Workshops (CVPR-WS)*.

Brust, Clemens-Alexander, Björn Barz, and Joachim Denzler (2021a). "Carpe Diem: A Lifelong Learning Tool for Automated Wildlife Surveillance". In: *Jahrestagung* *der Gesellschaft für Informatik Workshops (INFORMATIK-WS)*. DOI: `10.18420/informatik2021-034`.

Technical Reports and Pre-Prints

Brust, Clemens-Alexander, Sven Sickert, Marcel Simon, Erik Rodner, and Joachim Denzler (2016). *Neither Quick nor Proper – Evaluation of Quickprop for Learning Deep Neural Networks*. Tech. rep. Friedrich Schiller University Jena.

Brust, Clemens-Alexander, Bernd Gruner, and Tim Sonnekalb (2021). “ROMEO: Exploring Juliet through the Lens of Assembly Language”. In: arXiv: `2112.06623 [cs.CR]`.

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 eigener Prüfungsarbeiten ohne Kennzeichnung übernommen und habe alle von mir benutzten Hilfmittel, persönlichen Mittelungen 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.

Desweiteren 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 7. April 2022 Clemens-Alexander Brust