
Strategies for Practical Deep Learning

Doctoral Dissertation submitted to the
Faculty of Informatics of the Università della Svizzera Italiana
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

presented by
Lukas Tuggener

under the supervision of
Jürgen Schmidhuber and Thilo Stadelmann

11 2024

Dissertation Committee

Luca Maria Gambardella	Università della Svizzera Italiana, Switzerland
Rolf Krause	Università della Svizzera Italiana, Switzerland
Benjamin F. Grewe	Eidgenössische Technische Hochschule Zürich, Switzerland
Martin Jaggi	École Polytechnique Fédérale de Lausanne, Switzerland

Dissertation accepted on 29 11 2024

Signiert von:

E68CA477620F49F...

Research Advisor

Jürgen Schmidhuber

Signiert von:

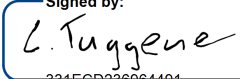
414F744C42464AF...

Co-Advisor

Thilo Stadelmann

PhD Program Director
The PhD program Director Walter Binder

I certify that except where due acknowledgement has been given, the work presented in this thesis is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; and the content of the thesis is the result of work which has been carried out since the official commencement date of the approved research program.

Signed by:

331E6D236964491...

Lukas Tuggener

Lugano, 29 11 2024

Abstract

In recent years, deep neural networks have achieved breakthrough results in diverse domains, from computer vision and natural language processing to game playing and life sciences. However, harnessing the full power of this technology in practical applications remains challenging. In this thesis, we explore strategies to address the challenges of applying deep learning to real-world pattern recognition problems. We tackle multiple practical problems, such as Optical Music Recognition (OMR), automated machine learning (AutoML), or the design of robust neural network architectures. In the context of OMR, we introduce two datasets, DeepScores and DeepScoresV2, the largest and most complete OMR datasets to date. Based on this data, we develop the first object detection method capable of handling the challenges of written music and methods to harden neural networks against the effects of degraded real-world data more than doubling detection performance on messy, degraded data. We then investigate the current state of AutoML, introduce a novel method for AutoML and extract design patterns for resource-constrained AutoML settings. In the latter parts of this thesis, we focus on the underlying issues that often cause neural networks to generalize poorly to real-world data. We first investigate the dataset dependency of modern CNN architectures. We show through an extensive empirical study that ImageNet alone is not sufficient to judge the power of CNN architectures and propose strategies for developing more universal evaluation methods. Finally, we tackle the lack of rotation invariance in modern vision systems and introduce a novel bio-inspired paradigm that significantly enhances the rotational robustness and outperforms the current state of the art by 19%.

Acknowledgements

I want to express my deepest gratitude to my supervisors Professors Jürgen Schmidhuber and Thilo Stadelmann, for guiding me throughout my academic journey and giving me the freedom to find my path while challenging me to become the best researcher I can be. Under their leadership, I was able to grow tremendously not only as a researcher but also as a person.

To my colleagues at CAI and InIT, I am very thankful for all the collaborations, discussions and the positive and stimulating environment they have created. Special thanks to Mohammadreza Amirian and Ismail Elezi for treading the treacherous waters of early PhD life with me.

Special thanks to Larissa for her love, patience, and support. She has been a tremendous source of strength for me. I thank her for standing by my side in all my hardest moments and celebrating the good ones with me. I am eternally grateful to have her by my side.

To my parents and sisters, I am thankful for their unconditional support. They provided me with the stable foundation upon which I was able to build and to this day give me a deep sense of calm and safety.

My dear friends Dominic, Pascal and Thomas I thank for their companionship, for all the laughs and heartfelt discussions we had throughout the years. They helped me get my mind off things even in the most stressful times.

Lastly, I want to thank all the healthcare workers who took care of me during the time when I could not.

Contents

Contents	vii
1 Introduction	1
1.1 Motivation	2
1.2 Research goal	3
1.3 Organization	4
1.4 List of contributions	5
1.4.1 First authorship	5
1.4.2 Co-authorship	6
2 Background	7
2.1 Strategies for practical NN design	7
2.2 Optical music recognition	8
3 Synthetic data as a strategy to overcome data scarcity: DeepScores	13
3.1 Introduction	13
3.2 <i>DeepScores</i> in the context of other datasets	14
3.2.1 Comparisons with computer vision datasets	17
3.2.2 Comparisons with OMR datasets	18
3.3 The <i>DeepScores</i> dataset	20
3.3.1 Quantitative properties	20
3.3.2 Flavors of ground truth	21
3.3.3 Dataset construction	23
3.4 Anticipated use and impact	24
3.4.1 Unique challenges	24
3.4.2 Towards next-generation computer vision	25
3.5 Discussion	26

4 Task-informed neural architecture design: The deep watershed detector	29
4.1 Introduction and problem statement	29
4.2 Related work	31
4.3 Deep watershed detection	32
4.3.1 Retrieving object centers	34
4.3.2 Object class and bounding box	34
4.3.3 Network architecture and losses	35
4.4 Experiments and results	36
4.4.1 Used datasets	36
4.4.2 Network training and experimental setup	37
4.4.3 Results	40
4.5 Discussion	42
5 Extending synthetic data with information and augmentation: DeepScoresV2	45
5.1 Introduction	46
5.2 The DeepScoresV2 dataset	47
5.2.1 Oriented bounding boxes	48
5.2.2 Extended symbol set	50
5.2.3 Additional features	50
5.3 Baseline results on DeepScoresV2	53
5.3.1 Reference experimental setup	53
5.3.2 Deep watershed detector	54
5.3.3 Faster R-CNN	54
5.3.4 Evaluation and discussion	54
5.4 The <i>RealScores</i> data	55
5.5 ScoreAug	57
5.6 Conclusion and future work	60
6 Automated machine learning under constrained resources	63
6.1 Introduction	63
6.2 Impact of practical machine learning	64
6.3 Survey of the current state of automated machine learning	65
6.4 Portfolio Hyperband	69
6.4.1 Intermediate conclusions	72
6.5 AutoDL for Vision and the sponge effect	72
6.5.1 Base architecture selection	73
6.5.2 Classifier design	74

6.5.3 The sponge effect in multiple modalities	76
6.5.4 Intermediate conclusions	77
7 Robust neural network design: Is it enough to optimize CNN architec-	
tures on ImageNet?	79
7.1 Introduction	80
7.2 Related work	82
7.3 Datasets	84
7.4 Experiments and results	85
7.4.1 Experimental setup	85
7.4.2 Experimental results	88
7.4.3 Impact of the number of classes	89
7.4.4 Identifying drivers of difference between datasets	91
7.5 Discussion and intermediate conclusions	94
8 Efficient rotation invariance in computer vision tasks: Artificial mental	
rotation	99
8.1 Introduction	100
8.2 Related work	101
8.3 Artificial mental rotation module	102
8.3.1 AMR module for CNNs	104
8.3.2 AMR module for ViTs	104
8.3.3 Motivation for add-on design	105
8.4 Experiments	105
8.5 Validity of self-supervised training built on artificial rotation	110
8.6 Application to a novel downstream task: semantic segmentation	113
8.7 Limitations and future work	113
8.8 Intermediate conclusions	114
9 Conclusions	115
9.1 Summary	115
9.2 Future directions	116
A Extended Result	119
A.1 Is it enough to optimize CNN architectures on Imagenet?	119
A.1.1 Verifying the numerical robustness of our study	119
A.1.2 Additional ablation studies	125
A.2 Mental Rotation	129
A.2.1 Investigating on Stanford Cars and Oxford Pet	129
A.2.2 Are the base network features useful for rotation estimation?	129

A.2.3 Re-digitized Images	132
B Artifacts	135
B.1 DeepScoresV2 and RealScores	135
B.2 Is it enough to optimize CNN architectures on Imagenet?	135
B.3 Mental Rotation	135
Bibliography	139
List of Figures	161
List of Tables	169

Chapter 1

Introduction

Humans and, to some extent, animals embody intelligent "programs" that allow us to assess situations around us, reason about the effect and usefulness of future actions, and even develop models and theories about the most fundamental rules of the environment we are inhabiting. At least since the inception of programmable computers, researchers have been intrigued by the prospect of designing intelligent machines (AI) that exhibit similar capabilities. Historically, AI research has explored many paradigms that span symbolic logic programming, expert systems, heuristic tree search, and connectionist approaches. In recent years, connectionist models such as Artificial Neural Networks (NNs) have revolutionized the way we approach pattern recognition in artificial intelligence (AI). NN-based systems have elevated the state of the art in a diverse array of academic fields such as computer vision [Cireşan, Meier, Masci, Gambardella and Schmidhuber, 2011b], machine translation [Wu et al., 2016] or game playing [Silver et al., 2017]. Much interest has been sparked by NN algorithms surpassing human performance on relevant, real-world tasks such as classification of traffic signs [Cireşan, Meier, Masci and Schmidhuber, 2012] or detecting cancer [Wang et al., 2016]. Fundamental to all of the above-mentioned advances is NN-based *deep learning* [Schmidhuber, 2015].

While NNs owe their name to the initial inspiration, the human brain, today's NNs are understood as arbitrary computational graphs. Each node (or neuron) computes an activation based on the outputs of upstream neurons or the inputs to the NN. The activations of the final nodes are the output of the NN. Each connection between nodes is assigned a real-valued weight determining the strength of said connection and collectively the weights determine the behaviour of the NN. Neural networks are usually separated into two classes, Feed Forward Neural Networks (FNNs) having an acyclic computation graph and Recurrent Neural

Networks (RNNs) that allow cycles. NNs encode a powerful family of functions that can approximate any continuous mapping up to an arbitrarily small error [Cybenko, 1989]. FNNs parametrize a single mapping, while RNNs can be interpreted as a program. A Neural Network layer typically consists of a learned linear mapping together with a fixed nonlinear activation function. The weights of a NN can efficiently be trained using gradient descent and error back propagation [Linnainmaa, 1970].

The term *deep learning* has been coined to describe NNs which consist of many stacked layers forming a "deep" computational graph. Networks based on this design paradigm are very powerful and have been responsible for many of the recent breakthroughs [Simonyan and Zisserman, 2015; Szegedy et al., 2015; Brown et al., 2020; Jumper et al., 2021]. Deep NNs can learn useful representations of the input data, which usually surpass the expressive power of handcrafted features. These properties make deep NNs highly attractive to academia and industry. However, certain detrimental properties of modern NNs as well as suboptimal structures in the current industrial landscape pose fundamental challenges that impede NN from being used to their fullest potential in many practical applications - overcoming these challenges will be the main focus of this thesis.

1.1 Motivation

Through interaction with and continued work on real-world pattern recognition problems faced by various industrial collaborators, we have identified a set of general *key challenges*. These regularly pose a significant obstacle when cutting-edge deep learning methods are supposed to be applied to novel real-world use cases. Since the same issues arise regularly and across highly diverse application domains, there is a need for generally applicable strategies rather than a common application-specific solution. We have identified the following challenges as the most impactful in terms of ubiquity and severity:

Data availability and quality: Deep NN are notoriously data hungry. Academic research is often centred around vast, cleanly annotated benchmark datasets such as ImageNet [Deng et al., 2009] for computer vision or SNLI [Bowman et al., 2015] for natural language processing. For many industrial applications, a large and clean enough dataset to train deep neural networks is not readily available.

Narrow model design: Scientific benchmarks often represent a narrow and well-defined problem. Researchers use these benchmarks for the development and testing of novel methods, which can lead to overly narrow engineered solutions that work very well for their intended purpose but do not generalize well to other

applications. This is especially true if the domain-leading benchmark consists of very particular data.

Demand for experts: Due to the fast-growing interest in deep learning within the academic and private sectors, there is a huge demand for experts in this field. This limited supply stunts adoption and, thus, the impact of deep learning outside of academia.

Fragility of deep neural networks: While undoubtedly powerful, deep neural networks are also very fragile and fail often and in spectacular ways when presented with data that is not properly represented in the training dataset.

1.2 Research goal

The research goal at the heart of this work is the development of suitable and generalizable strategies to tackle these issues on a fundamental level. This allows practitioners to bridge the gap between academic development and practical application and, therefore, further unlocks the potential of deep learning (beyond the cookie-cutter application of foundational models [Bommasani et al., 2021]) to a wider area of domains. Through preliminary studies and literature research, we have determined that the following strategies are particularly promising:

Domain shift-aware usage of synthetic data:(explored in Chapters 3 and 5) In the absence of sufficient training data, synthetic data [Nikolenko et al., 2021] can be generated using rendering engines, typesetting software or even neural networks themselves. Resulting issues of the domain shift between synthetic and real data can be addressed by input data augmentation [Sato et al., 2015], adversarial domain adaptation [Tzeng et al., 2017] or a combination thereof.

Task-informed neural architecture design:(explored in Chapter 4) Domains with little resemblance to academic benchmarks can cause state-of-the-art solutions to perform poorly. A possible remedy here is bespoke neural network design, which is carefully designed to perform well for the task at hand using a systematic approach.

Automated deep learning:(explored in Chapter 6) AutoDL methods [Zela et al., 2018] can be deployed fully automatically and tune themselves directly from the data. For hard problems, where fully automated deep learning fails, there exist methods which support human experts by automatically finding useful hyperparameters [Feurer and Hutter, 2019] or even candidate network architectures [Elsken et al., 2019].

Robust neural network design:(explored in Chapters 7 and 8) Most NN architecture research is focused on maximal predictive performance. Alternative design

goals such as robustness against adversarial attacks and stable performance across different domains can lead to less fragile architectures.

We systematically explored these strategies in the context of joint projects with industrial partners. Within these projects, we developed and implemented novel methods that enable the efficient use of deep neural networks in "messy" real-world conditions. Through this research process, we have learned which approaches for dealing with the above-mentioned challenges yield fruitful strategies and show how to apply them successfully.

1.3 Organization

This thesis is structured as follows: In Chapter 2, we give a brief overview of the most important background to our work.

In Chapter 3, we introduce DeepScores, a synthetic dataset for optical music recognition (OMR) containing 300'000 annotated sheets of music. With nearly a hundred million annotated small objects, it is by far the largest OMR dataset to date.

Chapter 4 introduces the Deep Watershed Detector, a novel object detection method. It is based on dense predictions and the deep watershed transformation and is designed to handle a large amount of small objects simultaneously.

Chapter 5 concludes our work OMR, here we present methods that enable the use of OMR methods in challenging real world scenarios. First, we introduce DeepScoresV2, an extension to DeepScores that contains more fine-grained label information, including oriented bounding boxes and higher-level rhythm and pitch information. Second, we show how advanced data augmentation can improve the performance of OMR on data of varying quality.

In Chapter 6 we discuss our contributions to the field of AutoML. We first introduce a novel hyperparameter optimization scheme we call portfolio hyperband second we discover and discuss an NN design paradigm we dub sponge effect.

In Chapter 7 we present an extensive study that challenges the notion that CNN architecture design solely based on ImageNet leads to generally effective convolutional neural network (CNN) architectures that perform well on a diverse set of datasets and application domains.

In Chapter 8 we introduce a novel, bio-inspired NN architecture that achieves efficient rotational invariance through artificial mental rotation.

Finally, in Chapter 9 we summarize our findings and conclude. Furthermore, do we give pointers towards worthwhile future work.

1.4 List of contributions

This section lists all of the peer-reviewed contributions we produced during the duration of the PhD research program.

1.4.1 First authorship

1. **Tuggener, L.**, Stadelmann, T. & Schmidhuber, J., (2023). Efficient Rotation Invariance in Deep Neural Networks through Artificial Mental Rotation. *Under review at AAAI 2025* (see Chapter [8](#)).
2. **Tuggener, L.**,^{*} Emberger, R.^{*}, Ghosh, A.^{*}, Sager, P.^{*}, Satyawar, Y. P., Montoya, J., Montoya, J., Goldschagg, S., Seibold, F., Gut, U., Ackermann, P., Schmidhuber, J., & Stadelmann, T. (2023). *Real world music object recognition*. *Transactions of the International Society for Music Information Retrieval* (see Chapter [5](#)).
3. **Tuggener, L.**, Schmidhuber, J., & Stadelmann, T. (2022). Is it enough to optimize CNN architectures on ImageNet? *In Frontiers in Computer Science*, 4, 1041703 (see Chapter [7](#)).
4. **Tuggener, L.**^{*}, Satyawar, Y. P.^{*}, Pacha, A., Schmidhuber, J., & Stadelmann, T. (2021). The DeepScoresV2 dataset and benchmark for music object detection. *In 2020 25th International Conference on Pattern Recognition (ICPR)* (pp. 9188-9195) IEEE (see Chapter [5](#)).
5. **Tuggener, L.**^{*}, Amirian, M.^{*}, Benites, F., von Däniken, P., Gupta, P., Schilling, F. P., & Stadelmann, T. (2020). Design patterns for resource-constrained automated deep-learning methods. *AI*, 1(4), 510-538 (see Chapter [6](#)).
6. **Tuggener, L.**, Amirian, M., Rombach, K., Lörwald, S., Varlet, A., Westermann, C., & Stadelmann, T. (2019). Automated machine learning in practice: state of the art and recent results. *In 2019 6th Swiss Conference on Data Science (SDS)* (pp. 31-36) IEEE (see Chapter [6](#)).
7. Elezi, I.^{*}, **Tuggener, L.**^{*}, Pelillo, M., & Stadelmann, T. (2018). DeepScores and Deep Watershed Detection: current state and open issues. *In 1st International Workshop on Reading Music Systems* (p. 13).

^{*} denotes equal contribution

8. **Tuggener, L.**, Elezi, I., Schmidhuber, J., & Stadelmann, T. (2018). Deep watershed detector for music object recognition. *In 19th International Society for Music Information Retrieval Conference (ISMIR)* (pp. 271-278) (see Chapter [4](#)).
9. **Tuggener, L.**, Elezi, I., Schmidhuber, J., Pelillo, M., & Stadelmann, T. (2018, August). Deepscores - a dataset for segmentation, detection and classification of tiny objects. *In 2018 24th International Conference on Pattern Recognition (ICPR)* (pp. 3704-3709) IEEE (see Chapter [3](#)).

1.4.2 Co-authorship

1. Emberger, R., Boss, J. M., Baumann, D., Seric, M., Huo, S., **Tuggener, L.**, Keller, E., & Stadelmann, T. (2023). Video object detection for privacy-preserving patient monitoring in intensive care. *In 10th IEEE Swiss Conference on Data Science (SDS)* IEEE.
2. Amirian, M., **Tuggener, L.**, Chavarriaga, R., Satyawon, Y. P., Schilling, F. P., Schwenker, F., & Stadelmann, T. (2021). Two to trust: AutoML for safe modelling and interpretable deep learning for robustness. *In Trustworthy AI-Integrating Learning, Optimization and Reasoning: First International Workshop, TAILOR 2020, Revised Selected Papers 1* (pp. 268-275) Springer International Publishing.
3. Amirian, M., Rombach, K., **Tuggener, L.**, Schilling, F. P., & Stadelmann, T. (2019). Efficient deep CNNs for cross-modal automated computer vision under time and space constraints. *In ECML-PKDD 2019*.
4. Stadelmann, T., Amirian, M., Arabaci, I., Arnold, M., Duivesteijn, G. F., Elezi, I., Geiger, M., Lörwald S., Meier, B., Rombach, K., & **Tuggener, L.** (2018). Deep learning in the wild. *In Artificial Neural Networks in Pattern Recognition: 8th IAPR TC3 Workshop, ANNPR 2018, Proceedings 8* (pp. 17-38) Springer International Publishing.

Chapter 2

Background

Section 2.1 gives an overview of previous attempts at extracting generalizable patterns for NN design. Section 2.2 introduces the field of optical music recognition (OMR), which is the main focus of Chapters 3 to 5. The remaining Chapters 6 to 8 are self-contained and feature all the necessary context information in their respective related works sections. We assume a solid understanding of and neural networks and deep learning, for a comprehensive overview of the field we suggest "Deep learning in neural networks: An overview" [Schmidhuber, 2015].

2.1 Strategies for practical NN design

It would be desirable to deduce NN design choices from an overarching theory [Poggio et al., 2018] with mathematical rigour. Unfortunately to this day, no such theory exists. Therefore researchers have focused on empirical methods to inform and validate their hypotheses. This focus on empiricism is often paired with the choice of one or a few landmark datasets such as ImageNet [Deng et al., 2009], MS CoCo [Lin et al., 2014] and the like, tending to lead to narrow and over-optimized solutions (more on this in Chapter 7).

There are a few strategies researchers have employed to develop methods that mark a “universal” progress and can be applied in various contexts. A straightforward way is designing extensive empirical studies aimed at distilling high-level “laws” about a class of NN rather than just one exact design. For example, the effect of data size on the generalization error in vision, text, and speech tasks has been studied and yielded consistent power-law behaviour in a large study [Hestness et al., 2017]. A more recent survey on the effect of model size, data size and training time of language models lead to several empirical scaling laws [Kaplan et al., 2020a]. Similar studies have been conducted in

the space of architecture search. EfficientNets [Tan and Le, 2019] leverage such insights to find architectures that are at the same time high-performing and resource-efficient. Here, it was observed that scaling width, depth, and input resolution have combined positive effects larger than scaling each factor in isolation.

Another top-down strategy is to identify certain structures or behaviours in human or animal intelligence and design methods with the goal of emulating them. Ma et al. [2022] introduce an effective and efficient computational framework built on the fundamental principals of parsimony and self consistency. Keurti et al. [2023] propose methods equipped with internal representations following a suitable homomorphism property enabling group representations of actions to keep internal consistency.

In this thesis, we have a strong focus on practical applicability with the overarching goal of extracting transferable strategies and patterns. Therefore we first systematically analyze our practical problems and identify what aspects require further research. We then employ various of the above-mentioned approaches to develop practically viable solutions to the concrete problems and simultaneously distil strategies with a wider application focus.

2.2 Optical music recognition

Music is one of the richest expressions of human art and culture. In order to conserve important pieces there is a rich tradition of written music dating back to at least 1250-1200 BC [West, 1994]. Currently, the most widely used notation for musical scores is the Common Western Music Notation (CWMN), see Figure 3.1 for an example. There are ongoing efforts to develop tools that can automatically read written music and transform it into a machine-readable format. This can save musical manuscripts from being lost to time and enable digital score management, such as finding a score with a specific melody from a library or automatic transposition. Out of these efforts, the research field of optical music recognition (OMR) has been formed [Rebelo et al., 2012; Calvo-Zaragoza et al., 2020].

With OMR having very similar overarching goals to Optical Character Recognition (OCR) for text the two fields appear very similar with many even calling OMR the "OCR for music". There are, however, some very significant differences between OMR and OCR. Understanding these differences is important for understanding the challenges and particularities of OMR. Music notation is of a compositional nature the pitch, duration and style of how a note is played can be influenced by a number of symbols other than the note itself, starting with the clef

and key signs at the beginning of a staff to accidentals and flags (among others) connected to the note itself. This can be further augmented by ties, tremolos or activation just to name a few. It is even the case that optically identical symbols can have different meanings depending on their context, see Section 3. This makes music notation far more complex to understand and reconstruct than text with its linear dependency between characters. The second major difference is that music notation is a two-dimensional signal, the vertical position on the staff has a profound impact on the meaning of any given symbol. A third important difference is the common occurrence of symbols that do not have a fixed shape or size, such as ties or slurs, that can even cross from one staff to the next.

Traditionally an OMR pipeline contains the the following steps:

Image preprocessing: In this first step, the input image has to be transformed into a form that is suitable for further processing. A crucial aspect of the preprocessing is the scaling such that the staves appear similar across all sheets. The desired scaling is often defined as the number of white pixels between two staff lines - the interline value. Additionally preprocessing often includes some form of de-skewing, filtering, noise removal and binarization [Fornés et al., 2005; Göcke, 2003; Fujinaga, 2004; dos Santos Cardoso et al., 2009].

Music object recognition:

In this step the individual music notation symbols are localized and classified. Most of our work concerns this area. Therefore, it is discussed in more detail below

Semantic music reconstruction and output generation: This step aims to extract the musical semantics from the detected symbols. This includes reconstructing all the necessary relationships between the objects and their position-specific interpretation. This information is then transformed into a desired machine-readable format. The most prominent approach is the introduction of a domain specific grammar [Bellini et al., 2008; Prearu, 1970]. Other approaches include the direct use of musical rules and heuristics [Droettboom et al., 2002] and abductive constraint logic programming [Ferrand et al., 1999]. A promising more recent solution is the definition of a relationship graph to encode all the necessary higher-level semantics between the found objects [Pacha et al., 2019]. Sequence models such as LSTM-RNNs have been shown to perform well for the retrieval of note-specific information, however, current methods are constrained to simple monophonic scores [Baró-Mas, 2017].

Music object recognition

A crucial sub-task of OMR is the localization and classification of individual symbols of music notation, also referred to as music object detection (MOR). A core difference between object detection in real-world photos and music object detection is the number of objects that usually appear in a single image. While there are usually a few objects in natural images, it is not uncommon to have hundreds or even thousands of objects of interest in a single music score image. Additionally, music symbols often rely heavily on the context to be classified correctly.

Classically, MOR has often been divided into a series of sub-steps. Each of these steps is performed by a specialized, usually hard-coded method.

Staff line detection and removal: First, the staff lines are localized and then isolated. A simple approach is to find maxima in horizontal pixel projections [Randriamahefa et al., 1993]. More sophisticated approaches employ Hough Transforms [MIYAO and NAKANO, 1996], vertical scan lines [Carter, 1989] or connected-component analysis [Fujinaga, 2004].

Symbol segmentation: In this second step, the individual music symbols are located and isolated. Usually, the image is decomposed according to the hierarchical structure of the written music, i.e. it is split up into individual stems to be processed individually. Then the atomic base symbols are extracted individually [Choudhury et al., 2000; Droettboom et al., 2002; Göcke, 2003]. This has been attempted using a plethora of approaches, including hand-crafted ones such as template matching [Reed and Parker, 1996], characteristics of projection features [Bellini et al., 2001], and features based on width height and topology [Choudhury et al., 2000].

Symbol recognition: Finally, the isolated symbols need to be classified. The most commonly used classifiers are neural networks [Bellini et al., 2001; MIYAO and NAKANO, 1996] and the k-nearest neighbours method [Choudhury et al., 2000; Fujinaga, 2004]. Others attempted to craft specific rules that incorporate musical rules [Roach and Tatem, 1988; Toyama et al., 2006].

In recent years, MOR research has heavily shifted its focus on deep learning methods. This shift simplified the MOR pipeline, requiring minimal preprocessing and performing MOR in a single step. A major roadblock to the adoption of deep learning to OMR was the lack of datasets of suitable size and quality. DeepScores as presented in Chapters 3 and 5 helped patching that gap. One of the pioneering works adapting deep learning to MOR is our development of the deep watershed detector architecture (see Chapter 4). Other contemporary work in that domain was the use of faster R-CNN [Pacha and Calvo-Zaragoza, 2018] and U-Net [Hajic Jr

[et al., 2018](#)] for MOR. Similar to many other fields of computer vision has the adoption of deep learning based methods brought a substantial step forward in MOR performance.

Chapter 3

Synthetic data as a strategy to overcome data scarcity: DeepScores

The DeepScores dataset with the goal of advancing the state-of-the-art in small objects recognition, and by placing the question of object recognition in the context of scene understanding is the focus of this chapter. DeepScores contains high quality images of musical scores, partitioned into 300,000 sheets of written music that contain symbols of different shapes and sizes. With close to a hundred millions of small objects, this makes our dataset not only unique, but also the largest public dataset. DeepScores comes with ground truth for object classification, detection and semantic segmentation. DeepScores thus poses a relevant challenge for computer vision in general, beyond the scope of optical music recognition (OMR) research. We present a detailed statistical analysis of the dataset, comparing it with other computer vision datasets like Caltech101/256, PASCAL VOC, SUN, SVHN, ImageNet, MS-COCO, smaller computer vision datasets, as well as with other OMR datasets. Finally, we provide baseline performances for object classification and give pointers to future research based on this dataset.

3.1 Introduction

Increased availability of data and computational power has often been followed by progress in computer vision and machine learning. The recent rise of deep learning in computer vision for instance has been promoted by availability of large image datasets [Deng et al., 2009] and increased computational power

This chapter is based on [Tuggener, Elezi, Schmidhuber, Pelillo and Stadelmann, 2018], previously published at ISMIR 2018 as a conference paper.

provided by GPUs [Oh and Jung, 2004; Raina et al., 2009; Cireřan et al., 2010; Cireřan, Meier, Masci, Gambardella and Schmidhuber, 2011a].

To the best of our knowledge, there are no Optical Music Recognition (OMR, see Section 2.2) systems yet that fully leverage the power of deep learning. We conjecture that this is caused in part by the lack of publicly available datasets of written music, big enough to train deep neural networks. The *DeepScores* dataset has been collected with OMR in mind, but addresses important aspects of next generation computer vision research that pertain to the size and number of objects per image.

Although there is already a number of clean, large datasets available to the computer vision community [Fei-Fei et al., 2004; Deng et al., 2009; Xiao et al., 2010; Everingham et al., 2010; Netzer et al., 2011; Lin et al., 2014], those datasets are similar to each other in the sense that for each image there are a few large objects of interest. Object detection approaches that have shown state-of-the-art performance under these circumstances, such as Faster R-CNN [Ren et al., 2014], SSD [Liu et al., 2016b] and YOLO [Redmon et al., 2016], demonstrate very poor off-the-shelf performances when applied to environments with large input images containing multiple small objects (see Section 3.4).

Sheets of written music, on the other hand, usually have dozens to hundreds of small salient objects. The class distribution of musical symbols is strongly skewed and the symbols have a large variability in size. Additionally, the OMR problem is very different from modern OCR [Goodfellow et al., 2013; Lee and Osindero, 2016]: while in classical OCR, the text is basically a 1D signal (symbols to be recognized are organized in lines of fixed height, in which they extend from left to right or vice versa), musical notation can additionally be stacked arbitrarily also on the vertical axis, thus becoming a 2D signal. This superposition property would exponentially increase the number of symbols to be recognized, if approached the usual way (which is intractable from a computational as well as from a classification point of view). It also makes segmentation very hard and does not imply a natural ordering of the symbols as for example in the SVHN dataset [Netzer et al., 2011].

3.2 DeepScores in the context of other datasets

DeepScores is a high quality dataset consisting of pages of written music, rendered at 400 dots per inch (dpi). It has 300'000 full pages as images, containing tens of millions of objects, separated in 118 classes. The purpose of the dataset is to facilitate general research on small object recognition, with direct applicability to

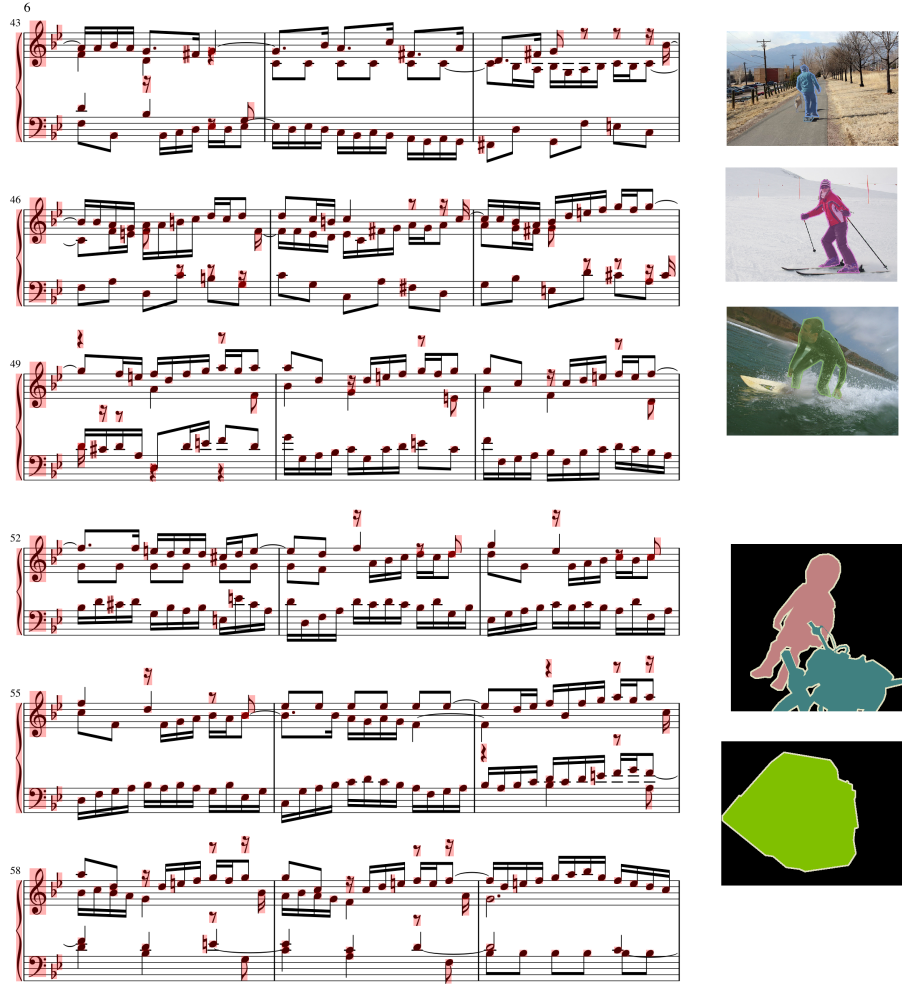


Figure 3.1. A typical image and ground truth from the DeepScores dataset (left), next to examples from the MS-COCO (3 images, top right) and PASCAL VOC (2 images, bottom right) datasets. Even though the music page is rendered at a much higher resolution, the objects are still smaller; the size ratio between the images is realistic despite all images being downscaled.

the recognition of musical symbols. We provide the dataset with three different kinds of ground truths (in the order of progressively increasing task complexity): object classification, semantic segmentation, and object detection.

Object classification in the context of computer vision is the procedure of annotating an image with a single label. Its recent history is closely linked to the success of deep convolutional learning models [Fukushima, 1980; Waibel, 1987; Zhang, 1988], leading to superhuman performance [Ciresan, Meier, Masci

and Schmidhuber, 2011; Ciresan, Meier, Masci, Gambardella and Schmidhuber, 2011a] and subsequent ImageNet object classification breakthroughs [Krizhevsky et al., 2012]. Shortly afterwards, similar systems achieved human-level accuracy also on ImageNet [Simonyan and Zisserman, 2015; Srivastava et al., 2015a; Szegedy et al., 2015; He et al., 2016]. Generally speaking, the ImageNet dataset [Deng et al., 2009] was a key ingredient for the success of image classification algorithms.

In *DeepScores*, we provide data for the classification task even though classifying musical symbols in isolation is not a challenging problem compared to classifying ImageNet images. But providing the dataset for classification, in addition to a neural network implementation that achieves high accuracy (see Section 3.4), might help to address the other two tasks. In fact, the first step in many computer vision models is to use a deep convolutional neural network pre-trained on ImageNet, and alter it for the task of image segmentation or image detection [Long et al., 2015; Ren et al., 2014]. We expect that the same technique can be used when it comes to detecting very small objects.

Semantic segmentation is the task of labeling each pixel of the image with one of the possible classes. State-of-the-art models are typically based on fully convolutional architectures [Ciresan, Giusti, Gambardella and Schmidhuber, 2012; Long et al., 2015]. The task arguably is a significantly more difficult problem than image classification, with the recent success being largely attributed to the release of high quality datasets like PASCAL VOC [Everingham et al., 2010] and MS-COCO [Lin et al., 2014].

In *DeepScores*, we provide ground truth for each pixel in all the images, having roughly 10^{12} labeled pixels in the dataset. In the next section, we compare these figures with existing datasets.

Object detection is by far the most interesting and challenging task: to classify all the objects in the image, and at the same time to find their precise position in the image. State-of-the-art algorithms are pipeline convolutional models, typically having combined cost functions for detection and classification [Ren et al., 2014; Liu et al., 2016b; Redmon et al., 2016]. The task can be combined with segmentation, which means that the algorithm is required to provide masks (instead of bounding boxes) for each of the objects in the image [He et al., 2017]. It differs from mere segmentation because the result shows which pixel collections form individual objects. Similar to the case of semantic segmentation above, the PASCAL VOC and especially MS-COCO datasets have played an important role in the recent success of object detection algorithms.

In *DeepScores*, we provide bounding boxes and labels for each of the musical symbols in the dataset. With around 80 million objects, this makes our dataset

the largest one released so far, and highly challenging: the above-mentioned algorithms did not work well on our dataset in preliminary comprehensive experiments. We attribute this to the fact that most of the models used for object detection are fitted to datasets which have few but large objects. On the contrary, our dataset contains a lot of very small objects, which means that new models might need to be created in order to deal with it.

3.2.1 Comparisons with computer vision datasets

Compared with some of the most used datasets in the field of computer vision, *DeepScores* has by far the largest number of objects, in addition of having the highest resolution. In particular, images of *DeepScores* have a resolution of $1'894 \times 2'668$ pixels, which is at least four times higher than the resolutions of datasets we compare with. Table 3.1 contains quantitative comparisons of *DeepScores* with other datasets, while the following paragraphs bring in also qualitative aspects.

SVHN, the street view house numbers dataset [Netzer et al., 2011], contains 600'000 labeled digits cropped from street view images. Compared to *DeepScores*, the number of objects in SVHN is two orders of magnitude lower, and the number of objects per image is two to three orders of magnitude lower.

ImageNet contains a large number of images and (as a competition) different tracks (classification, detection and segmentation) that together have proven to be a solid foundation for many computer vision projects. However, the objects in ImageNet are quite large, while the number of objects per image is very small. Unlike ImageNet, *DeepScores* tries to address this issue by going to the other extreme, providing a very large number of very small objects, with images having significantly higher resolution than all the other mentioned datasets.

PASCAL VOC is a dataset which has been assembled mostly for the tasks of detection and segmentation. Compared to ImageNet, the dataset has slightly more objects per image, but the number of images is comparatively small: our dataset is one order of magnitude bigger in the number of images, and three orders of magnitude bigger in the number of objects.

MS-COCO is a large upgrade over PASCAL VOC on both the number of images and number of objects per image. With more than 300K images containing more than 3 millions of objects, the dataset is very useful for various tasks in computer vision. However, like ImageNet, the number of objects per image is still more than one order of magnitude lower than in our dataset, while the objects are relatively large.

A number of other datasets have been released during the years, which have contributed to the progress of the field, and some of them have been used for

different competitions. **MNIST** [LeCun, Cortes and Burges, 1998] is the first “large” dataset in the fields of machine learning and computer vision. It has tens of thousands of 28x28 pixels grayscale images, each containing a handwritten digit. The dataset is a solved classification problem and during the last decade has been used mostly for prototyping new models. Nowadays, this is changing with more challenging datasets like **CIFAR-10/CIFAR-100** [Krizhevsky et al., 2009] being preferred. Similar to MNIST, those datasets contain an object per image (32x32 color pixels), which do not make them ideal for more challenging problems like detection and segmentation.

Caltech-101/Caltech-256 [Gregory et al., 2007] are more interesting datasets considering that both the resolution and the number of images are larger. Still, the images contain only a single object, making them only useful for the process of image classification. **SUN** [Xiao et al., 2010] is a scene understanding dataset, containing over 100k images, each labeled with a single class.

The online and offline Chinese handwriting databases, **CASIA-OLHWDB** and **CASIA-HWDB** [Liu et al., 2011], were produced by 1’020 writers using a digital pen on paper, such that both online and offline data were obtained. The samples include both isolated characters and handwritten texts (continuous scripts). Both datasets have millions of samples, separated into 7’356 classes, making them far more interesting and challenging than digit datasets.

The German traffic sign recognition benchmark (**GTSRB**) is a multi-category classification competition held at IJCNN 2011 [Stallkamp et al., 2011]. The corresponding dataset comprises a comprehensive collection of more than 50’000 lifelike traffic sign images, reflecting the strong variations in visual appearance of signs due to distance, illumination, weather conditions, partial occlusions, and rotations. The dataset has 43 classes with unbalanced class frequencies.

3.2.2 Comparisons with OMR datasets

Several OMR datasets have been released in the past with a specific focus on the computer music community. *DeepScores* will be of use both for general computer vision as well as to the OMR community (compare Section 3.4).

3.2.2.1 Handwritten scores

The Handwritten Online Musical Symbols dataset **HOMS** [Calvo-Zaragoza and Oncina, 2014] is a reference corpus with around 15’000 samples for research on the recognition of online handwritten music notation. For each sample, the

Dataset	#classes	#images	#objects	#pixels
MNIST	10	70k	70k	55m
CIFAR-10	10	60k	60k	61m
CIFAR-100	100	60k	60k	61m
Caltech-101	101	9k	9k	700m
Caltech-256	256	31k	31k	2b
SUN	397	17k	17k	6b
PASCAL VOC	21	10k	30k	2.5b
MS COCO	91	330k	3.5m	100b
ImageNet	200	500k	600k	125b
SVHN	10	200k	630k	4b
CASIA online	7356	5090	1.35	nn
CASIA offline	7356	5090	1.35m	nn
GTSRB	43	50k	50k	nn
<i>DeepScores</i>	118	300k	80m	1.5t

Table 3.1. Information about the number of classes, images and objects for some of the most common used datasets in computer vision. The number of pixels is estimated due to most datasets not having fixed image sizes. We used the SUN 2012 object detection specifications for SUN, and the statistics of ILSVRC 2014 [Russakovsky et al., 2015] detection task for ImageNet.

individual strokes that the musician wrote on a Samsung Tablet using a stylus were recorded and can be used in online and offline scenarios.

The **CVC-MUSCIMA** database [Fornes et al., 2012] contains handwritten music images, which have been specially designed for writer identification and staff removal tasks. The database contains 1'000 music sheets written by 50 different musicians with characteristic handwriting styles.

MUSICMA++ [Jan Hajič and Pecina, 2017] is a dataset of handwritten music for musical symbol detection that is based on the MUSCIMA dataset. It contains 91'255 written symbols, consisting of both notation primitives and higher-level notation objects, such as key signatures or time signatures. There are 23'352 notes in the dataset, of which 21'356 have a full notehead, 1'648 have an empty notehead, and 348 are grace notes.

The **Capitan Collection** [Calvo-Zaragoza et al., 2016] is a corpus collected via an electronic pen while tracing isolated music symbols from early manuscripts. The dataset contains information on both the sequence followed by the pen (capitan stroke) as well as the patch of the source under the tracing itself (capitan

score). In total, the dataset contains 10'230 samples unevenly spread over 30 classes.

3.2.2.2 Print quality scores

The **MuseScore Monophonic MusicXML Dataset** [van der Wel and Ullrich, 2017] is one of the largest OMR dataset to date, consisting of 7'000 monophonic scores. While the dataset has high quality images, it doesn't resemble real-world musical scores which are not monophonic and thus have many lines per image.

Further OMR datasets of printed scores are reviewed by the **OMR-Datasets** project¹. *DeepScores* is by far larger than any of these or the above-mentioned dataset, containing more images and musical symbols than all the other datasets combined. In addition, *DeepScores* contains only real-world scores (i.e., symbols in context as they appear in real written music), while the other datasets are either synthetic or reduced (containing only symbols in isolation or just a line per image). The sheer scale of *DeepScores* makes it highly usable for the modern deep learning algorithms. While convolutional neural networks have been used before for OMR [van der Wel and Ullrich, 2017], *DeepScores* for the first time enables the training of very large and deep models.

3.3 The DeepScores dataset

3.3.1 Quantitative properties

DeepScores contains around 300'000 pages of digitally rendered music scores and has ground truth for 118 different symbol classes. The number of labeled music symbol instances is roughly 80 million (4-5 orders of magnitudes higher than in the other music datasets; when speaking of symbols, we mean labeled musical symbols that are to be recognized as objects in the task at hand). The number of symbols on one page can vary from as low as 4 to as high as 7'664 symbols. On average, a sheet (i.e., an image) contains around 243 symbols. Table 3.2 gives the mean, standard deviation, median, maximum and minimum number of symbols per page in the second column.

Another interesting aspect of *DeepScores* is the class distribution (see Figure 3.3). Obviously, some classes contain more symbols than other classes (see also Table 3.2, column 3). It can be seen that the average number of elements per

¹See <https://apacha.github.io/OMR-Datasets/>.

Statistic	Symbols per sheet	Symbols per class
Mean	243	650k
Std. dev.	203	4m
Maximum	7'664	44m
Minimum	4	18
Median	212	20k

Table 3.2. Statistical measures for the occurrences of symbols per musical sheet and per class (rounded).

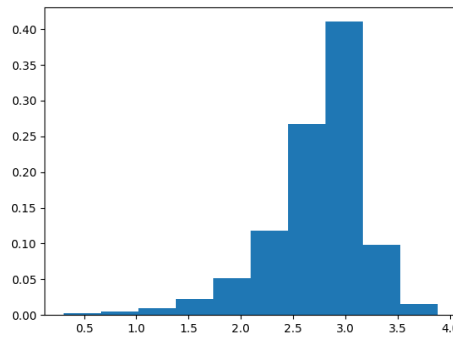


Figure 3.3. Histogram for the distribution of symbols over all images (logarithmic scale on abscissa, ordinate weighted to give unit area). The majority of images contain from 100 to 1000 objects.

the 80m symbols as a single patch for classification purposes, but constrain the dataset for this simpler task to a random subset of reasonable size (see Section 3.4). The patches have a size of 45 x 170 and contain the full original context of the symbol (i.e., they are cropped out of real world musical scores). Each patch is centered around the symbol's bounding box (see Figure 3.4d).

For object detection, there is an accompanying XML file for each image in *DeepScores*. The XML file has an object node for each symbol instance present on the page, which contains class and bounding box coordinates.

For semantic segmentation, there is an accompanying PNG file for each image. This PNG has identical size as the initial image, but each pixel has been recolored to represent the symbol class it is part of. As in Figure 3.4c, the background is white, with the published images using grayscale colors from 0 to 118 for ease of use in the softmax layer of potential models.



(a) Snippet of an input image.



(b) Bounding boxes rendered over single objects from snippet 3.4a for object detection.



(c) Color-based pixel level labels (the differences can be hard to see, but there is a distinct color per symbol class) for semantic segmentation.

(d) Patches centered around specific symbols (in this case: `gClef`) for object classification.

Figure 3.4. Examples for the different flavors of ground truth available in DeepScores.

3.3.3 Dataset construction

DeepScores is constructed by synthesizing from a large collection of written music in a digital format: crowd-sourced MusicXML files publicly available from MuscScore² and used by permission. The rendering of MuscXML with accompanying ground truth for the three flavors of granularity is done by a custom software using the SVG back-end of the open-source music engraving software LilyPond. The rendered SVG files not only contain all the musical symbols, but also additional tags that allow for identifying what musical symbol each SVG path belongs to.

To achieve a realistic variety in the data even though all images are digitally rendered and therefore have perfect image quality, five different music fonts have

²<https://musescore.com>



Figure 3.5. The same patch, rendered using five different fonts.

been used for rendering (see Figure 3.5). Python scripts finally extract the three types of ground truth from this basis of SVG data and save the images as PNG using the CairoSVG library.

A key feature of a dataset is the definition of the classes to be included. Due to their compositional nature, there are many ways to define classes of music symbols: is it for example a “c” note with duration 8 (`noteheadBlack`) or is it a black notehead (`noteheadBlack`) and a flag (`flag8thUp` or `flag8thDown`)? Adding to this complexity, there is a huge number of special and thus infrequent symbols in music notation. The selected set is the result of many discussions with music experts and contains the most important symbols. We decided to use atomic symbol parts as classes which makes it possible for everyone to define composite symbols in an application-dependent way.

3.4 Anticipated use and impact

3.4.1 Unique challenges

One of the key challenges this dataset poses upon modeling approaches is the sheer amount of objects on a single image. Two other properties of music notation impose challenges: First, there is a big variability in object size as can be seen for example in Figure 3.6. Second, music notation has the special feature that context matters: two objects having identical appearance can belong to a different class depending on the local surroundings (see Figure 3.7). To our knowledge there is no other freely available large scale dataset that shares this trait.

Moreover, datasets like ImageNet are close to being perfectly balanced, with the number of images/objects per class being a constant. This clearly isn’t the case with the *DeepScores* dataset, where the most common class contains more than half of the symbols in the dataset, and the top 10% of classes contain more than 85% of the symbols in the entire dataset. This extremely skewed distribution

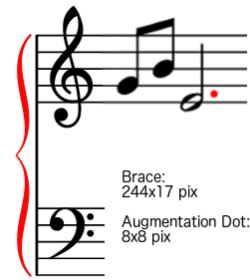


Figure 3.6. The possible size difference of objects in music notation, illustrated by `brace` and `augmentationDot`.

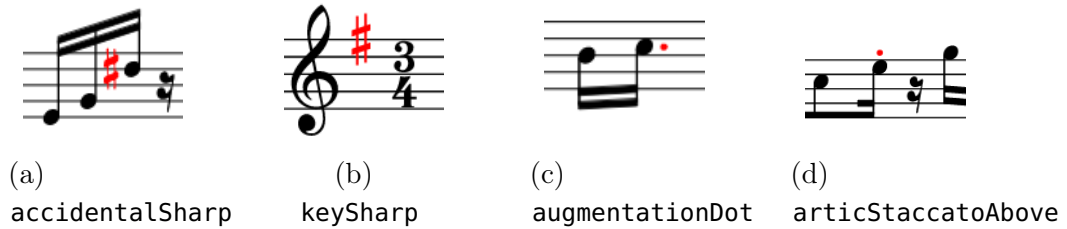


Figure 3.7. Examples for the importance of context for classifying musical symbols: in both rows, the class of otherwise similar looking objects changes depending on the surrounding objects.

resembles many real-world cases for example in anomaly detection and industrial quality control.

3.4.2 Towards next-generation computer vision

Classifying the musical symbols in *DeepScores* should nevertheless not be a problem: all these symbols have very clear black and white borders, their shape has limited variability and they are rendered at very high resolution (see Figure 3.2). Due to these reasons we assumed that classification on *DeepScores* should be a relatively easy task, given CNNs usually deal well with these kinds of objects. To support this assumption, we fitted a simple residual-CNN [He et al., 2016] (a special case of Highway Net [Srivastava et al., 2015a]) with 25 convolutional layers and about 8 million trainable parameters. Using the Adam optimizer [Kingma and Ba, 2015] with the hyper-parameters proposed by the authors, we reached an accuracy of over 0.98 in just ten epochs. This shows that classification will

indeed not be a big issue and CNNs are able to deal with labels that not only depend on an object but also its surroundings.

Detection, however, is more challenging: we evaluated SSD's and YOLO's fitness for the detection task on *DeepScores* and applied Faster R-CNN - with very little success. We conjecture that one of the main problems is that these region proposal-based systems seem to become computationally overwhelmed for this type of data, due to the sheer number of proposals necessary to find the many small objects.

Both observations - easy classification but challenging detection - lie at the heart of what we think makes *DeepScores* very useful: it offers the challenging scenario of many tiny objects that cannot be approached using current datasets (see Section 3.2). On the other hand, *DeepScores* is probably the easiest scenario of that kind, because classifying single musical objects is relatively easy and the dataset contains a vast amount of training data. *DeepScores* thus is a prime candidate to develop next generation computer vision methods that scale to many tiny objects on large images: many real-world problems deal with high-resolution images, with images containing hundreds of objects and with images containing very small objects in them. This might be automated driving and other robotics use cases, medical applications with full-resolution imaging techniques as data sources, or surveillance tasks e.g. in sports arenas and other public places.

Finally, *DeepScores* will be a valuable source for pre-training models: transfer learning has been one of the most important ingredients in the advancement of computer vision. The first step in many computer vision models [Long et al., 2015; Ren et al., 2014] is to use a deep convolutional neural network pre-trained on ImageNet, and alter it for the task of image segmentation or object detection, or use it on considerably smaller, task-dependent final training sets. *DeepScores* will be of value specifically in the area of OMR, but more generally to allow the development of algorithms that focus on the fine-grained structure of smaller objects while simultaneously being able to scale to many objects of that nature.

3.5 Discussion

We have presented the conception and creation of *DeepScores* - the largest publicly and freely available dataset for computer vision applications in terms of image size and contained objects. Compared to other well-known datasets, *DeepScores* has large images (more than four times larger than the average) containing many (one to two orders of magnitude more) very small (down to a few pixels, but varying by several orders of magnitude) objects that change their class belonging

depending on the visual context. The dataset is made up of sheets of written music, synthesized from the largest public corpus of MusicXML. It comprises ground truth for the tasks of object classification, semantic segmentation and object detection.

We have argued that the unique properties of *DeepScores* make the dataset suitable for use in the development of general next generation computer vision methods that are able to work on large images with tiny objects. This ability is crucial for real-world applications like robotics, automated driving, medical image analysis or surveillance, besides OMR. We have motivated that object classification is relatively easy on *DeepScores*, making it therefore the potentially cheapest way to work on a challenging detection task. We thus expect impact on general object detection algorithms.

A weakness of the *DeepScores* dataset is that all the data is digitally rendered. Linear models (or piecewise linear models like neural networks) have been shown to not generalize well when the distribution of the real-world data is far from the distribution of the dataset the model has been trained on [Torralba and Efros, 2011; Szegedy et al., 2013]. Future work on the dataset will include developing and publishing scripts to perturb the data in order to make it look more like real (scanned) written music, and evaluation of the transfer performance of models trained on *DeepScores*.

Future work *with* the dataset will - besides the general impact predicted above - directly impact OMR: the full potential of deep neural networks is still to be realized on musical scores as shown in the next Chapter.

Chapter 4

Task-informed neural architecture design: The deep watershed detector

In this chapter, we introduce a novel object detection method, based on synthetic energy maps and the watershed transform, called Deep Watershed Detector (DWD). Our method is specifically tailored to deal with high-resolution images that contain a large number of very small objects and is therefore able to process full pages of written music. We present state-of-the-art detection results of common music symbols and show DWD's ability to work with synthetic scores equally well as on handwritten music.

4.1 Introduction and problem statement

The goal of Optical Music Recognition (OMR) is to transform images of printed or handwritten music scores into machine readable form, thereby understanding the semantic meaning of music notation [Bainbridge and Bell, 2001] (See Section 2.2). The two main challenges of OMR are: first the accurate detection and classification of music objects in digital images; and second, the reconstruction of valid music in some digital format. This work focuses solely on the first task, meaning that we recover the position and class (based on the shape only) of every object without inferring any higher-level information.

Recent progress in computer vision [Girshick et al., 2018] thanks to the adaptation of convolutional neural networks (CNNs) [Fukushima and Miyake, 1982; LeCun, Boser, Denker, Henderson, Howard, Hubbard and Jackel, 1989] pro-

This chapter is based on [Tugener, Elezi, Schmidhuber and Stadelmann, 2018], previously published at ICPR 2018 as a conference paper.

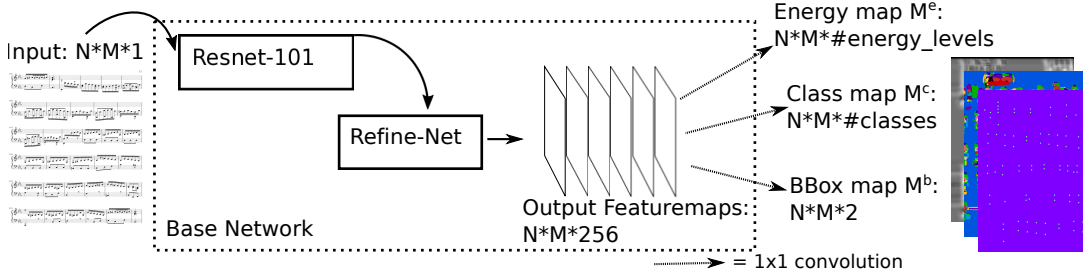


Figure 4.1. Illustration of the DWD network and its sub-components together with input and outputs. The outputs have been cropped to improve visibility

vide a solid foundation for the assumption that OMR systems can be drastically improved by using CNNs as well. Initial results of applying deep learning [Schmidhuber, 2015] to heavily restricted settings such as staffline removal [Sánchez and Calvo-Zaragoza, 2017], symbol classification [Pacha and Eidenberger, 2017] or end-to-end OMR for monophonic scores [Calvo-Zaragoza et al., 2017], support such expectations.

In this chapter, we introduce a novel general object detection method called Deep Watershed Detector (DWD) motivated by the following two hypotheses: a) deep learning can be used to overcome the classical OMR approach of having hand-crafted pipelines of many preprocessing steps [Rebelo et al., 2010] by being able to operate in a fully data-driven fashion; b) deep learning can cope with larger, more complex inputs than simple glyphs, thereby learning to recognize musical symbols in their context. This will disambiguate meanings (e.g., between staccato and augmentation dots) and allow the system to directly detect a complex alphabet.

DWD operates on full pages of music scores in one pass without any preprocessing besides interline normalization and detects handwritten and digitally rendered music symbols without any restriction on the alphabet of symbols to be detected. We further show that it learns a meaningful representation of music notation and achieves state-of-the-art detection rates on common symbols.

The remaining structure of this chapter is as follows: Section 4.2 puts our approach in context with existing methods; in Section 4.3 we derive our original end-to-end model, and give a detailed explanation on how we use the deep watershed transform for the task of object recognition; Section 4.4 reports on experimental results of our system on the *DeepScores* digitally rendered dataset in addition to the *MUSCIMA++* handwritten dataset before we conclude in Section 4.5 with a discussion of our findings.

4.2 Related work

The visual detection and recognition of objects is one of the most central problems in the field of computer vision. With the recent developments of CNNs, many competing CNN-based approaches have been proposed to solve the problem. R-CNNs [Girshick et al., 2014], and in particular their successors [Ren et al., 2014], are generally considered to be state-of-the-art models in object recognition, and many developed recognition systems are based on R-CNN. On the other hand, researchers have also proposed models which are tailored towards computational efficiency instead of detection accuracy. YOLO systems [Redmon and Farhadi, 2017] and Single-Shot Detectors [Liu et al., 2016a] while slightly compromising on accuracy, are significantly faster than R-CNN models, and can even achieve super real-time performance.

A common aspect of the above-mentioned methods is that they are specifically developed to work on cases where the images are relatively small, and where images contain a small number of relatively large objects [Everingham et al., 2010; Lin et al., 2014]. On the contrary, musical sheets usually have high-resolution, and contain a very large number of very small objects, making the mentioned methods not suitable for the task.

The watershed transform is a well understood method that has been applied to segmentation for decades [Beucher et al., 1992]. Bai and Urtasun [Bai and Urtasun, 2017] were first to propose combining the strengths of deep learning with the power of this classical method. They proposed to directly learn the energy (in our application the distance to an object center) for the watershed transform such that all dividing ridges are at the same height. As a consequence, the components can be extracted by a cut at a single energy level without leading to over-segmentation. The model has been shown to achieve state of the art performance on object segmentation.

For the most part, OMR detectors have been rule-based systems working well only within a hard set of constraints [Rebelo et al., 2010]. Typically, they require domain knowledge, and work well only on simple typeset music scores with a known music font, and a relatively small number of classes [Rossant and Bloch, 2007]. When faced with low-quality images, complex or even handwritten scores [Baro et al., 2016], the performance of these models quickly degrades, to some degree because errors propagate from one step to another [Pacha and Eidenberger, 2017]. Additionally, it is not clear what to do when the classes change, and in many cases, this requires building the new model from scratch.

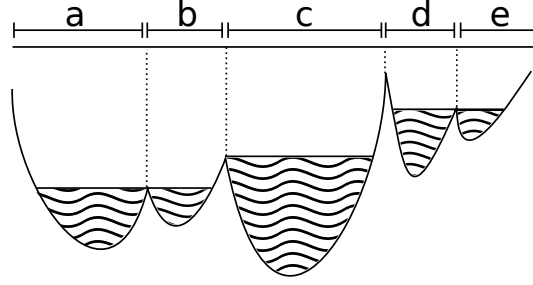
In response to the above mentioned issues some deep learning based, data driven approaches have been developed. Hajic and Pecina [Jr. and Pecina, 2017]

proposed an adaptation of Faster R-CNN with a custom region proposal mechanism based on the morphological skeleton to accurately detect noteheads, while Choi et al. [Choi et al., 2017] were able to detect accidentals in dense piano scores with high accuracy, given previously detected noteheads, that are being used as input-features to the network. A big limitation of both approaches is that the experiments have been done only on a tiny vocabulary of the musical symbols, and therefore their scalability remains an open question.

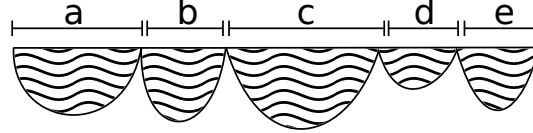
To our knowledge, the best results so far has been reported in the work of Pacha and Choi [Pacha, Choi, Coüasnon, Ricquebourg and Zanibbi, 2018] where they explored many models on the *MUSCIMA++* [Jan Hajič and Pecina, 2017] dataset of handwritten music notation. They got the best results with a Faster R-CNN model, achieving an impressive score on the standard mAP metric. A serious limitation of that work is that the system was not designed in an end-to-end fashion and needs heavy pre- and post-processing. In particular, they cropped the images in a context-sensitive way, by cutting images first vertically and then horizontally, such that each image contains exactly one staff and has a width-to-height-ratio of no more than 2 : 1, with about 15% horizontal overlap to adjacent slices. In practice, this means that all objects significantly exceeding the size of such a cropped region will neither appear in the training nor testing data, as only annotations that have an intersection-over-area of 0.8 or higher between the object and the cropped region are considered part of the ground truth. Furthermore, all the intermediate results must be combined to one concise final prediction, which is a non-trivial task.

4.3 Deep watershed detection

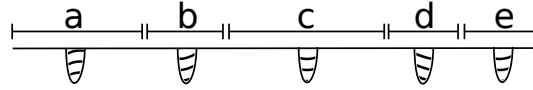
In this section, we present the Deep Watershed Detector (DWD) as a novel object detection system, built on the idea of the deep watershed transform [Bai and Urtasun, 2017]. The watershed transform [Beucher et al., 1992] is a mathematically well understood method with a simple core idea that can be applied to any topological surface. The algorithm starts filling up the surface from all the local minima, with all the resulting basins corresponding to connected regions. When applied to image gradients, the basins correspond to homogeneous regions of said image (see Fig. 4.2a). One key drawback of the watershed transform is its tendency to over segment. This issue can be addressed by using the deep watershed transform. It combines the classical method with deep learning by training a deep neural network to create an energy surface based on an input image. This has the advantage that one can design the energy surface to have



(a) One-dimensional energy function of five classes without any structural constraints.



(b) Energy function for the same five classes with fixed boundary energy.



(c) Energy function for the same five classes this time with small energy markers at the class centers.

Figure 4.2. Illustration of the watershed transform applied to different one-dimensional functions.

certain properties. When designed in a way that all segmentation boundaries have energy zero, the watershed transform is reduced to a simple cutoff at a fixed energy level (see Fig. 4.2b). An objectness energy of this fashion has been used by Bai and Urtasun for instance segmentation [Bai and Urtasun, 2017]. Since we want to do object detection, we further simplify the desired energy surface to having small conical energy peaks of radius n pixels at the center of each object and be zero everywhere else (see Fig. 4.2c).

More formally, we define our energy surface (or: energy map) M^e as follows:

$$M_{(i,j)}^e = \max \begin{cases} \operatorname{argmax}_{c \in C} [E_{\max} \cdot (1 - \frac{\sqrt{(i-c_i)^2 + (j-c_j)^2}}{r})] \\ 0 \end{cases} \quad (4.1)$$

where $M_{(i,j)}^e$ is the value of M^e at position (i, j) , C is the set of all object centers and c_i, c_j are the center coordinates of a given center c . E_{\max} corresponds to the maximum energy and r is the radius of the center marking.

At first glance this definition might lead to the misinterpretation that object centers that are closer together than r cannot be disambiguated using the water-

shed transform on M^e . This is not the case since we can cut the energy map at any given energy level between 1 and E_{max} . However, using this method it is not possible to detect multiple bounding boxes that share the exact same center.

4.3.1 Retrieving object centers

After computing an estimate \hat{M}^e of the energy map, we retrieve the coordinates of detected objects by the following steps:

1. Cut the energy map at a certain fixed energy level and then binarize the result.
2. Label the resulting connected components, using the two-pass algorithm [Wu et al., 2009]. Every component receives a label l in $1...n$, for every component o^l we define O_{ind}^l as the set of all tuples (i, j) for which the pixel with coordinates j and i is part of o^l .
3. The center \hat{c}^l of any component o^l is given by its center of gravity:

$$\hat{c}^l = o_{center}^l = |O_{ind}^l|^{-1} \cdot \sum_{(i,j) \in O_{ind}^l} (i, j) \quad (4.2)$$

We use these component centers \hat{c} as estimates for the object centers c .

4.3.2 Object class and bounding box

In order to recover bounding boxes we do not only need the object centers, but also the object classes and bounding box dimensions. To achieve this we output two additional maps M^c and M^b as predictions of our network. M^c is defined as:

$$M_{(i,j)}^c = \begin{cases} \Lambda_{(i,j)}, & \text{if } M_{(i,j)}^e > 0 \\ \Lambda_{background}, & \text{otherwise} \end{cases} \quad (4.3)$$

where $\Lambda_{background}$ is the class label indicating background and $\Lambda_{(i,j)}$ is the class label associated with the center c that is closest to (i, j) . We define our estimate for the class of component o^l by a majority vote of all values $\hat{M}_{(i,j)}^c$ for all $(i, j) \in O_{ind}^l$, where \hat{M}^c is the estimate of M^c . Finally, we define the bounding box map M^b as follows:

$$M_{(i,j)}^b = \begin{cases} (y^l, x^l), & \text{if } M_{(i,j)}^e > 0 \\ (0, 0), & \text{otherwise} \end{cases} \quad (4.4)$$

where y^l and x^l are the width and height of the bounding box for component o^l . Based on this we define our bounding box estimation as the average of all estimations for label l :

$$(\hat{y}^l, \hat{x}^l) = |O_{ind}^l|^{-1} \cdot \sum_{(i,j) \in O_{ind}^l} \hat{M}_{(i,j)}^b \quad (4.5)$$

4.3.3 Network architecture and losses

As mentioned above we use a deep neural network to predict the dense output maps M^e , M^c and M^b (see Fig. 4.1). The base neural network for this prediction can be any fully convolutional network with the same input and output dimensions. We use a ResNet-101 [He et al., 2016] (a special case of a Highway Net [Srivastava et al., 2015b]) in conjunction with the elaborate RefineNet [Lin et al., 2017] upsampling architecture. For the estimators defined above it is crucial to have the highest spacial prediction resolution possible. Our network has three output layers, all of which are an 1 by 1 convolution applied to the last feature map of the RefineNet.

4.3.3.1 Energy prediction

We predict a quantized and one-hot encoded version of M^e , called M^{eo} , by applying a 1 by 1 convolution of depth E_{max} to the last feature map of the base network. The loss of the prediction \hat{M}^{eo} , $loss^e$, is defined as the cross-entropy between M^{eo} and \hat{M}^{eo} .

4.3.3.2 Class prediction

We again use the corresponding one-hot encoded version M^{co} and predict it using an 1 by 1 convolution, with the depth equal to the number of classes, on the last feature map of the base network. The cross-entropy $loss^c$ is calculated between M^{co} and \hat{M}^{co} . Since it is not the goal of this prediction to distinguish between foreground and background, all the loss stemming from locations with $M^e = 0$ will get masked out.

4.3.3.3 Bounding box prediction

M^b is predicted in its initial form using an 1 by 1 convolution of depth 2 on the last feature map of the base network. The bounding box loss $loss^b$ is the mean-squared difference between M^b and \hat{M}^b . For $loss^b$, the components stemming from background locations will be masked out analogous to $loss^c$.

4.3.3.4 Combined prediction

We want to jointly train in all tasks, therefore we define a total loss $loss^{tot}$ as:

$$loss^{tot} = w_1 * \frac{loss^e}{v^e} + w_2 * \frac{loss^c}{v^c} + w_3 * \frac{loss^b}{v^b} \quad (4.6)$$

where the v are running means of the corresponding losses and the scalars w are hyper-parameters of the DWD network. We purposefully use very short extraction heads of one convolutional layer; by doing so we force the base network to do all three tasks simultaneously. We expect this leads to the base network learning a meaningful representation of music notation, from which it can extract the solutions of the three above defined tasks.

4.4 Experiments and results

4.4.1 Used datasets

For our experiments we use two datasets: *DeepScores* [Tuggener, Elezi, Schmidhuber, Pelillo and Stadelmann, 2018] and *MUSCIMA++* [Jan Hajič and Pecina, 2017].

DeepScores is currently the largest publicly available dataset of musical sheets with ground truth for various machine learning tasks, consisting of high-quality pages of written music, rendered at 400 dots per inch. The dataset has 300,000 full pages as images, containing tens of millions of objects, separated in 123 classes. We randomly split the set into training and testing, using 200k images for training and 50k images each for testing and validation. The dataset being so large allows efficient training of large convolutional neural networks, in addition to being suitable for transfer learning [Yosinski et al., 2014].

MUSCIMA++ is a dataset of handwritten music notation for musical symbol detection. It contains 91,255 symbols spread into 140 pages, consisting of both notation primitives and higher-level notation objects, such as key signatures or time signatures. It features 105 object classes. There are 23,352 notes in the dataset, of which 21,356 have a full notehead, 1,648 have an empty notehead, and 348 are grace notes. We randomly split the dataset into training, validation, and testing, with the training set consisting of 110 pages, while validation and testing each consists of 15 pages.

4.4.2 Network training and experimental setup

We pre-train our network in two stages in order to achieve reasonable results. First we train the ResNet on music symbol classification using the *DeepScores* classification dataset [Tuggener, Elezi, Schmidhuber, Pelillo and Stadelmann, 2018]. Then, we train the ResNet and RefineNet jointly on semantic segmentation data also available from *DeepScores*. After this pre-training stage, we are able to use the network on the tasks defined above in Section 4.3.3.

Since music notation is composed of hierarchically organized sub-symbols, there does not exist a canonical way to define a set of atomic symbols to be detected (e.g., individual numbers in time signatures vs. complete time signatures). We address this issue using a fully data-driven approach by detecting atomic classes as they are provided by the two datasets.

We rescale every input image to the desired interline value. We use 10 pixels for *DeepScores* and 20 pixels for *MUSCIMA++*. Other than that we apply no preprocessing. We do not define a subset of target objects for our experiments, but attempt to detect all classes for which there is ground truth available. We always feed single images to the network, i.e. we only use batch size = 1. During training we crop the full page input (and the ground truth) to patches of 960 by 960 pixels using randomized coordinates. This serves two purposes: it saves GPU memory and performs efficient data augmentation. This way the network never sees the exact same input twice, even if we train for many epochs. For all of the results described below we train individually on $loss^e$, $loss^c$ and $loss^b$ and then refine the training using $loss^{tot}$. It turns out that the prediction of M^e is the most fragile to effects introduced by training on the other losses, therefore we retrain on $loss^e$ again after training on the individual losses in the order defined above, before moving on to $loss^{tot}$. All the training is done using the RMSProp optimizer [Tieleman and Hinton, 2012] with a learning rate of 0.001 and a decay rate of 0.995.

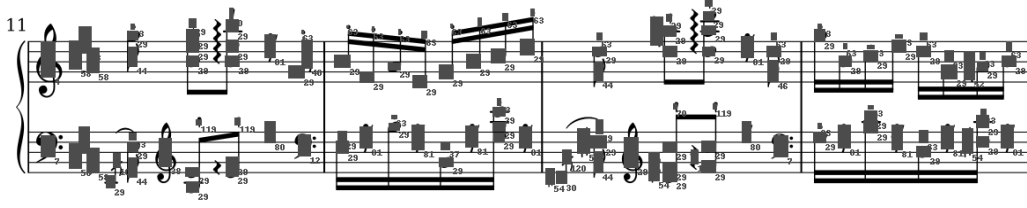
Since our design is invariant to how many objects are present on the input (as long as their centers do not overlap) and we want to obtain bounding boxes for full pages at once, we feed whole pages to the network at inference time. The maximum input size is only bounded by the memory of the GPU. For typical pieces of sheet music, this is not an issue, but pieces that use very small interline values (e.g. pieces written for conductors) result in very large inputs due to the interline normalization. At about 10.5 million pixels even a Tesla P40 with 24 gigabytes runs out of memory.

Class	AP@ $\frac{1}{2}$	Class	AP@ $\frac{1}{4}$
rest16th	0.8773	tuplet6	0.9252
noteheadBlack	0.8619	keySharp	0.9240
keySharp	0.8185	rest16th	0.9233
tuplet6	0.8028	noteheadBlack	0.9200
restQuarter	0.7942	accidentalSharp	0.8897
rest8th	0.7803	rest32nd	0.8658
noteheadHalf	0.7474	noteheadHalf	0.8593
flag8thUp	0.7325	rest8th	0.8544
flag8thDown	0.6634	restQuarter	0.8462
accidentalSharp	0.6626	accidentalNatural	0.8417
accidentalNatural	0.6559	flag8thUp	0.8279
tuplet3	0.6298	keyFlat	0.8134
noteheadWhole	0.6265	flag8thDown	0.7917
dynamicMF	0.5563	tuplet3	0.7601
rest32nd	0.5420	noteheadWhole	0.7523
flag16thUp	0.5320	fClef	0.7184
restWhole	0.5180	restWhole	0.7183
timeSig8	0.5180	dynamicPiano	0.7069
accidentalFlat	0.4949	accidentalFlat	0.6759
keyFlat	0.4685	flag16thUp	0.6621

Table 4.1. AP with overlap 0.5 and overlap 0.25 for the twenty best detected classes of the DeepScores dataset.

Class	AP@ $\frac{1}{2}$	Class	AP@ $\frac{1}{4}$
half-rest	0.8981	whole-rest	0.9762
flat	0.8752	ledger-line	0.9163
natural	0.8531	half-rest	0.8981
whole-rest	0.8226	flat	0.8752
notehead-full	0.8044	natural	0.8711
sharp	0.8033	stem	0.8377
notehead-empty	0.7475	staccato-dot	0.8302
stem	0.7426	notehead-full	0.8298
quarter-rest	0.6699	sharp	0.8121
8th-rest	0.6432	tenuto	0.7903
f-clef	0.6395	notehead-empty	0.7475
numeral-4	0.6391	duration-dot	0.7285
letter-c	0.6313	numeral-4	0.7158
letter-c	0.6313	8th-flag	0.7055
8th-flag	0.6051	quarter-rest	0.6849
slur	0.5699	letter-c	0.6643
beam	0.5188	letter-c	0.6643
time-signature	0.4940	8th-rest	0.6432
staccato-dot	0.4793	beam	0.6412
letter-o	0.4793	f-clef	0.6395

Table 4.2. AP with overlap 0.5 and overlap 0.25 for the twenty best detected classes from MUSCIMA++.



(a) Example result from DeepScores with detected bounding boxes as overlays. The tiny numbers are class labels from the dataset introduced with the overlay. This system is roughly one-fourth of the size of a typical DeepScores input we process at once.



(b) Example result from MUSCIMA++ with detected bounding boxes and class labels as overlays. This system is roughly one-half of the size of a typical processed MUSCIMA++ input. The images are random picks amongst inputs with many symbols.

Figure 4.3. Detection results for DeepScores and MUSCIMA++ examples, drawn on crops from corresponding input images.

4.4.3 Results

Table 4.1 shows the average precision (AP) for the twenty best detected classes with an overlap of the detected bounding box and ground truth of 50% and 25%, respectively. We observe that in both cases there are common symbol classes that get detected very well, but there is also a steep fall off. The detection rate outside the top twenty continues to drop and is almost zero for most of the rare classes. We further observe that there is a significant performance gain for the lower overlap threshold, indicating that the bounding-box regression is not very accurate.

Fig. 4.3 shows an example detection for qualitative analysis. It confirms the conclusions drawn above. The rarest symbol present, an arpeggio, is not detected at all, while the bounding boxes are sometimes inaccurate, especially for large objects (note that stems, bar-lines and beams are not part of the *DeepScores* alphabet and hence do not constitute missed detections). On the other hand, staccato dots are detected very well. This is surprising since they are typically hard to detect due to their small size and the context-dependent interpretation of

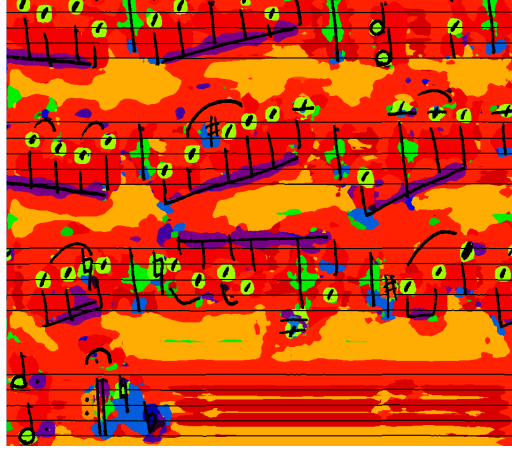


Figure 4.4. Estimate of a class map \hat{M}^c for every input pixel with the corresponding MUSCIMA++ input overlayed.

the symbol shape (compare the dots in dotted notes or F-clefs). We attribute this to the opportunity of detecting objects in context, enabled by training on larger parts of full raw pages of sheet music in contrast to the classical processing of tiny, pre-processed image patches or glyphs.

The results for the experiments on *MUSCIMA++* in Tab. 4.2 and Fig. 4.3b show a very similar outcome. This is intriguing because it suggests that the difficulty in detecting digitally rendered and handwritten scores might be smaller than anticipated. We attribute this to the fully data-driven approach enabled by deep learning instead of hand-crafted rules for handling individual symbols. It is worth noting that ledger-lines are detected with very high performance (see $AP@_{\frac{1}{4}}$). This explains the relatively poor detection of note-heads on *MUSCIMA++*, since they tend to overlap.

Fig. 4.4 shows an estimate for a class map with its corresponding input overlayed. Each color corresponds to one class. This figure proves that the network is learning a sensible representation of music notation: even though it is only trained to mark the centers of each object with the correct colors, it learns a primitive segmentation mask. This is best illustrated by the (purple) segmentation of the beams.

4.5 Discussion

We have presented a novel method for object detection that is specifically tailored to detect many tiny objects on large inputs. We have shown that it is able to detect common symbols of music notation with high precision, both in digitally rendered music as well as in handwritten music, without a drop in performance when moving to the "more complicate" handwritten input. This suggests that deep learning based approaches are able to deal with handwritten sheets just as well as with digitally rendered ones, additionally to their benefit of recognizing objects in their context and with minimal preprocessing as compared to classical OMR pipelines. Pacha et al. [Pacha, Choi, Coüasnon, Ricquebourg and Zanibbi, 2018] show that higher detection rates, especially for uncommon symbols, are possible when using R-CNN on small snippets (cp. Fig. 4.5). Despite their higher scores, it is unclear how recognition performance is affected when results of overlapping and potentially disagreeing snippets are aggregated to full page results. A big advantage of our end-to-end system is the complete avoidance of error propagation in longer recognition pipeline of independent components like classifiers, aggregators, etc [LeCun, Bottou, Bengio and Haffner, 1998]. Moreover, our full-page end-to-end approach has the advantages of speed (compared to a sliding window patch classifier), change of domain (we use the same architecture for both the digital and handwritten datasets) and is easily integrated into complete OMR frameworks.

Arguably the biggest problem we faced is that symbol classes in the dataset are heavily unbalanced. In the *DeepScores* dataset in particular, the class *notehead* contains more than half of all the symbols in the entire dataset, while the top 10 classes contain more than 85% of the symbols. Considering that we did not do any class-balancing whatsoever, this imbalance had its effect in training. We observe that in cases where the symbol is common, we get a very high average precision, but it quickly drops when symbols become less common. Furthermore, it is interesting to observe that the neural network actually forgets about the existence of these rarer symbols: Fig. 4.6 depicts the evolution of $loss^b$ of a network that is already trained and gets further trained for another 8,000 iterations. When faced with an image containing rare symbols, the initial loss is larger than the loss on more common images. But to our surprise, later during the training process, the loss actually increases when the net encounters rare symbols again, giving the impression that the network is actually treating these symbols as outliers and ignoring them.

Recently, researchers have been investigating the ability of state-of-the-art recognition systems on more challenging (compared to the usual *PASCAL VOC*



Figure 4.5. Typical input snippet used by Pacha et al. [Pacha, Choi, Coüasnon, Ricquebourg and Zanibbi, 2018]

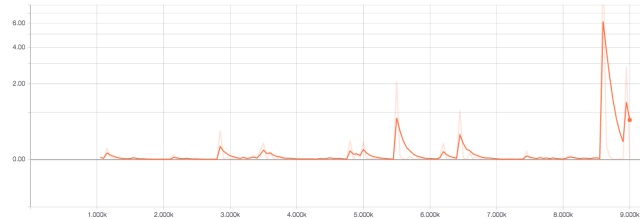


Figure 4.6. Evolution of $loss^b$ (on the ordinate) of a sufficiently trained network, when training for another 8000 iterations (on the abscissa).

[Everingham et al., 2010] and *MS-COCO* [Lin et al., 2014]) natural datasets, like *DOTA* [Xia et al., 2017], and unsurprisingly, the results leave much to be desired. The *DOTA* dataset shares a lot of similarities with musical datasets, with images being high resolution and containing hundreds of small objects, making it a suitable benchmark for our DWD method to recognize tiny objects.

Chapter 5

Extending synthetic data with information and augmentation: DeepScoresV2

In this chapter, we present three contributions that facilitate the use of OMR in messy, real-world settings. First, we present DeepScoresV2, an extended version of the DeepScores dataset for optical music recognition (OMR). We improve upon the original DeepScores dataset by providing much more detailed annotations, namely (a) annotations for 135 classes, including fundamental symbols of non-fixed size and shape, increasing the number of annotated symbols by 23%; (b) oriented bounding boxes; (c) higher-level rhythm and pitch information (onset beat for all symbols and line position for noteheads) These additions open up the potential for future advancement in OMR research. Additionally, we release two state-of-the-art baselines for DeepScoresV2 based on Faster R-CNN and the Deep Watershed Detector. An analysis of the baselines shows that regular orthogonal bounding boxes are unsuitable for objects which are long, small, and potentially rotated, such as ties and beams, which demonstrates the need for detection algorithms that naturally incorporate object angles. Second, we introduce RealScores a small test set consisting of manually annotated pages of varying real-world quality, sourced from the Petrucci Music Library Lastly, we present ScoreAug a sophisticated input augmentation scheme that can reduce the gap between sanitised benchmarks and

This chapter is based on Tuggener et al. [2021], previously published at ICPR 2020 as a conference paper and excerpts of Tuggener et al. [2023], previously published in Transactions of the International Society for Music Information Retrieval as a journal paper. This was a highly collaborative work where no clear boundary of contributions can be drawn, however, the excerpts focus on the areas where the author of this thesis focused on the most.

realistic tasks through a combination of synthetic data and noisy perturbations of real-world documents. For the access to the data, code and pre-trained models please refer to the Artifacts section [B](#) in the Appendix.

5.1 Introduction

DeepScoresV2 is an extended and improved version of the original DeepScores dataset that specifically addresses these issues and makes the following contributions: we (a) add 20 formerly absent classes including symbols without fixed size or shape but are nonetheless fundamental to music notation, thereby increasing the list of musical symbols that can be detected by 23%; (b) add ground truth for oriented bounding boxes, thus enabling research into detectors with potentially much higher precision; (c) add ground truth for further higher-level musical semantics, therefore making the dataset valuable for tasks beyond pure music object detection downstream in OMR; (d) add a compatibility mode for DeepScoresV2 and MUSCIMA++ such that the two datasets can be easily used in conjunction; and (e) provide pre-trained state-of-the-art detectors and benchmarking results for comparisons.

The original DeepScores dataset was designed with only fixed-shape symbols in mind. In DeepScoresV2, the available classes additionally include variably-shaped symbols, such as beams and slurs, which can be as small as spanning two closely neighboring objects to being as wide as the entire page, as shown in Figure [5.2](#). This makes DeepScoresV2 not only more complete, but also makes achieving a high precision much more challenging.

Musical notation contains symbols that tend to have a very high width-to-height ratio and are non-orthogonal to the image axes. This leads to orthogonal bounding boxes that contain a large number of background pixels (see Figure [5.1](#), left) and, even more problematic, to bounding boxes that overlap largely with other bounding boxes (see Figure [5.1](#), right).

To address this issue, DeepScoresV2 contains both orthogonal bounding boxes as well as oriented bounding boxes that cover the minimum rectangular area around each object, as illustrated in Figure [5.1](#) (yellow). This ensures that bounding boxes represent their corresponding objects more accurately and reduce the amount of overlap with other objects. A quantitative analysis shows that DeepScoresV2 indeed represents symbols more accurately through its oriented bounding boxes that cover on average 13.34% less background.

Due to the high complexity of musical notation, OMR datasets generally have different, non-compatible annotations. This makes working on different datasets

very complex and laborious. To enable easy interoperability with MUSCIMA++, we ship DeepScoresV2 with a compatibility mode that allows for out-of-the-box mixing of the two datasets. This is desirable for increased diversity.

We present baseline object detection algorithms on DeepScoresV2 which show that while some symbols in the extended symbol set can be detected reasonably well with state-of-the-art models, future work will be needed to achieve good detection on all, especially the new symbols. Nevertheless, the baseline models and the experimental setup used are ready to use for any future study for comparisons.

The enhancements added into DeepScoresV2 enable the training of fine-grained and sophisticated MOR algorithms. However, it still consists of synthetic data which makes it a bad test set to judge the performance of the MOR system in the real-world. Therefore we created Realscores, a small-scale, handlabeled test set consisting of 14 sheets real-world data. The data has been sourced from the Petrucci Music Library and selected with care to contain a wide variety of perturbations common in re-digitized physical data.

To bridge this domain gap between synthetic and realistic images, we propose ScoreAug, which creates augmentations for diversity in feature distributions during training. It uses real-world, scanned blank pages with natural signs of degradation and combines them with the synthetic input from our initial dataset. By mending those two together, realistic-looking samples can be created on-the-fly. The result of that operation is that the trained models generalise better to real-world data.

5.2 The DeepScoresV2 dataset

Object detectors that are pre-trained on natural images result in poor performance when used for music object detection [Tuggener, Elezi, Schmidhuber, Pelillo and Stadelmann, 2018]. This is due to the following challenges:

- Large scale (size) variations both between different classes of symbols and between different instances of a single class. For example, some symbols, like slurs, are dynamically sized according to their contextual meanings while maintaining the same class.
- A large number of symbols on each page of sheet music. Typically, most object detection datasets contain in the range of tens to hundreds of instances per image. In contrast, most sheet music pages contain between a few hundred to thousands of individual objects per page.

- Many very thin symbols, which are not aligned with the axes of the image. This causes orthogonal bounding boxes to become an imprecise representation of musical symbols, containing more background than foreground pixels in each bounding box.

To address these challenges, we present DeepScoresV2, a large-scale, high-quality, fully annotated optical musical recognition dataset. DeepScoresV2 consists of 255,386 pages of digitally engraved sheet music, rendered at 400 dots per inch (DPI) with tens of millions of symbols. We also provide a dense version of this dataset consisting of 1,714 of the most diverse and challenging images split into 1,362 training images and 352 test images. Annotations are also provided with the option of using multiple category names to allow for compatibility with the MUSCIMA++ dataset [Jan Hajič and Pecina, 2017]. This is done so that cross-modal validation of techniques could be performed on both printed and handwritten music scores. Finally, we have excluded those pages from DeepScoresV2 that have malformed annotations in DeepScores to reduce the chances that incorrectly labeled annotations would appear in DeepScoresV2. Images are provided as PNG files along with segmentations in indexed PNG files, instance segmentation in PNG files, and annotations in JSON files.

5.2.1 Oriented bounding boxes

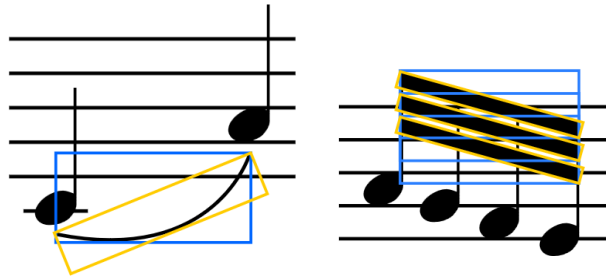


Figure 5.1. Two examples of slanted symbols with their corresponding orthogonal bounding boxes (blue) and oriented bounding boxes (yellow). Orthogonal bounding boxes contain a significant amount of background pixels (left) and can have ample overlap with other bounding boxes (right). Oriented bounding boxes reduce these issues.

One of the main new features of DeepScoresV2 are the oriented bounding boxes. The area outlined by an orthogonal bounding box often contains a

Class Name	Class average background pixel ratio (%)		
	Orthogonal BBox	Oriented BBox	Improvement
slur	92.69	86.30	6.89
tie	84.48	78.83	6.69
clef8	77.73	54.03	30.49
beam	35.40	11.73	66.86
noteheadBlack	25.98	17.00	34.57
rest16th	66.05	54.98	16.76
Overall	55.83	49.26	13.34

Table 5.1. Average background area reduction for selected classes and average overall reduction. “Background pixel ratio” shows what percentage of pixels within a bounding box is part of the background rather than the foreground.

significant amount of background pixels, especially when the respective symbol is thin and slanted like a beam or slur. To address these shortcomings, we have added oriented bounding boxes to DeepScoresV2 labelled as 8-tuples $[x_0, y_0, x_1, y_1, x_2, y_2, x_3, y_3]$. These bounding boxes are always rectangular, but generally at an angle relative to the image axes, and calculated from the minimum area rectangle around each object instance as follows: using the PNG pixel array and the original DeepScores annotations that contain orthogonal bounding box information for every symbol, we calculate the oriented bounding box by treating each pixel of a symbol within the orthogonal bounding box as a point in a 2D space and effectively turn the problem into finding the minimum area rectangle around this set of points. This is finally calculated using the minimum area rectangle function provided in the Shapely package¹.

Qualitatively, these oriented bounding boxes are better representations of their objects as they more clearly depict the shape of the object, as seen in Figure 5.1. Quantitatively, we can reduce the number of background pixels contained within a bounding box by an average of 13.34%. A detailed analysis of some prominent classes is depicted in Table 5.1.

For easy use of this bounding box scheme, we are making available the OB-BAnns toolkit² as a framework-agnostic tool to work with the DeepScoresV2 dataset. It provides abstractions to load annotations, get image-annotation pairs by index or image ID, visualize the dataset, and calculate validation metrics,

¹<https://github.com/Toblerity/Shapely>

²https://github.com/yvan674/obb_anns

with the most computationally intensive operations implemented in C++. The toolkit can also be used to work with any dataset containing both oriented bounding boxes as well as ground-truth segmentation. The data schema and further instructions on how to use the toolkit can be found in the respective repository.

5.2.2 Extended symbol set

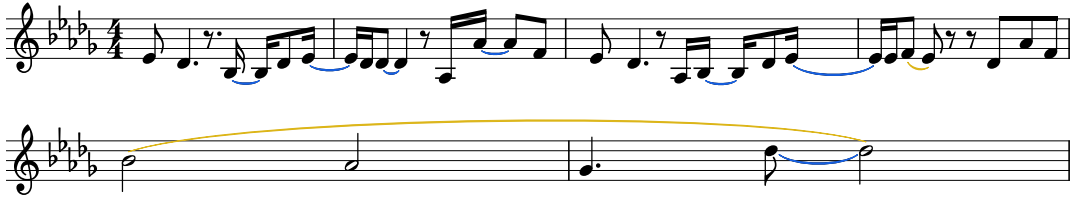


Figure 5.2. Slurs (yellow) and ties (blue) can vary significantly in size, ranging from relatively short instances (top) to almost as wide as the entire staff (bottom). Depicted music are excerpts of “You make it real” by James Morrison.

DeepScoresV2 introduces an extended symbol set encompassing variably sized symbols, including some changes for added musical context and having a few name changes to become more self-consistent (see Table 5.2 for a detailed overview). By incorporating variably sized symbols, a richer musical representation can be extracted as opposed to using the original set of classes, which contains only fixed-sized symbols. Symbols such as slurs and ties, which may span from two neighboring notes and up an entire line of music, as seen in Figure 5.2, are particularly difficult for machine learning algorithms to understand as a single class due to their scale variability. Newly introduced symbols from this category are beams, dynamicDiminuendoHairpins, dynamicCrescendoHairpins, slurs, stems, and ties.

New contextual symbols are also introduced as part of the extended symbol set, namely the stem, tuplet, tupletBracket, ottavaBrackets, ledgerLines, and tremolo classes. Finally, some symbol names are changed to be more consistent: for example, compound dynamic symbols and time signatures have all been reduced to their components and clef names have been rectified to be in line with dynamics, flags, and rests.

5.2.3 Additional features

Apart from the aforementioned major contributions, there are numerous smaller additions included in DeepScoresV2:

Symbol	Change with respect to DeepScores
beam	Added
clef	Changed all symbols to use clefX naming scheme and removed "changed" suffix
staff	Added
hairpin	Added dynamicDiminuendoHairpin and dynamicCrescendoHairpin
dynamics	Changed to individual symbols, e.g. dynamicS, dynamicF, dynamicZ
ledgerLine	Added
noteheads	Added InSpace and OnLine suffixes
ottavaBracket	Added
restHNr	Added
restLonga	Removed
restMaxima	Removed
slur	Added
stem	Added
tie	Added
timeSig	Changed to individual numerals, e.g. timeSig0, timeSig1
tremolo	Added tremolo0 - tremolo5
tuplet	Added tuplet1 - tuplet9
tupletBracket	Added

Table 5.2. Summary of changes to symbol classes in DeepScoresV2. Most notable is the addition of hairpins, beams, slurs, and ties. Additionally, some names have been changed to become more consistent, and certain compound symbols have been split into their component symbols for added robustness. Classes that do not occur in the dataset have been removed.

5.2.3.1 Cross-dataset compatibility

Compatibility between OMR datasets has long been neglected, which has made it very difficult to compare different approaches and re-use existing work. To alleviate this problem, we define a compatibility mode that allows us to jointly use the MUSCIMA++ and DeepScoresV2 datasets, e.g., for model training or evaluation. The MUSCIMA++ dataset was chosen because it is, to the best of our knowledge, the only large OMR dataset which contains annotations on a similar level. Furthermore, the underlying material - handwritten music scores in modern notation - is a great complement for the DeepScoresV2 dataset. Compatibility is enforced by (a) confining the symbol sets to the subset of classes that appear in both datasets; (b) choosing a decomposition of musical symbols into detectable objects that both datasets can provide; and (c) aligning the class names wherever possible by following the SMuFL [\[Spreadbury and Piéchaud, 2015\]](#) conventions.

5.2.3.2 Staff information

DeepScoresV2 introduces additional information regarding the position of the notes with respect to the staff to facilitate pitch recognition. All notehead classes are split into `-InSpace` and `-OnLine` sub-classes, making subsequent position-based pitch detection more robust against minor perturbations. For direct staff detection, every note head in DeepScoresV2 has its relative staff position stored in its annotation as an additional field.

5.2.3.3 Onset information

To enable research of OMR models with a deeper musical understanding, DeepScoresV2 also contains annotation for temporal onset for every symbol (on which beat a given symbol starts). This allows for the training of models capable of much higher level reconstruction of the music than just localization and classification of individual objects.

5.2.3.4 Instance segmentation annotations

While DeepScores already contains pixel-wise semantic segmentation ground truth, DeepScoresV2 ships with additional *instance* segmentation masks. We provide instance segmentation in separate PNG files containing instance information in the RGB-channels, starting from 1 and reset with every page. The instance number is encoded in the hexadecimal color value used (e.g. instance 1 has a

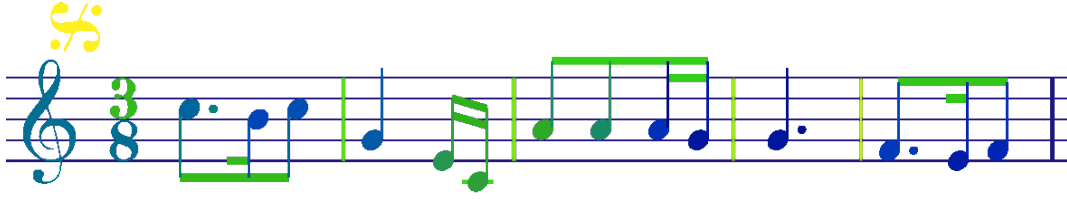


Figure 5.3. An example of the instance segmentation ground truth. Every symbol occurrence has its own color due to the encoding of instance information as color values. Color differences have been exaggerated for better readability.

color value of #000001). An example of an instance segmentation mask is shown in Figure 5.3.

5.3 Baseline results on DeepScoresV2

To highlight some of the peculiarities of the dataset as well as to enable future work, we have created a reference experimental setup and trained and evaluated two baseline models.

5.3.1 Reference experimental setup

The results presented in this section are obtained by training the models using the train split of DeepScoresV2 dense until the training loss saturates. Previous experiments showed that due to training on random crops and the huge number of symbols, overfitting is not an issue. Both models are trained on the aligned (non-oriented) bounding boxes because there is currently no established method for oriented object detection in the OMR field. The results are reported using the metrics Average Precision at an overlap of 0.5 (AP0.5) [Everingham et al., 2010] and COCO mean Average Precision (mAP) [Lin et al., 2014], computed by the evaluation function of the OBBAnns toolkit. Detailed information on the hyperparameters of the individual models are contained in the configuration files of the respective codebases.

5.3.2 Deep watershed detector

As a first baseline, we provide the Deep Watershed Detector (DWD) [Tuggener, Elezi, Schmidhuber and Stadelmann, 2018] that represents the current state of the art on DeepScores. We train it without any major modification from its originally published design. The only change is that we use the data at full resolution instead of applying a scaling factor of 0.5. Due to the large image size, this requires training on crops and also to perform inference using a crop and reassemble process that involves multiple forward passes. However, this can be done in a straight forward fashion since DWD is built entirely on fully convolutional neural networks [Long et al., 2015]. For this baseline, we disable staff and ledger line detection because the DWD is by design unable to detect objects that share the same object centers.

5.3.3 Faster R-CNN

For the second baseline, we chose the Faster R-CNN architecture [Girshick, 2015], based on the model published in Pacha et al. [Pacha, Hajič jr. and Calvo-Zaragoza, 2018] that features specifically designed anchor boxes. As a backbone to the model, we use the newly introduced HRNet [Sun et al., 2019], which is able to produce extremely high-resolution features. This combination in itself is novel to the field of OMR.

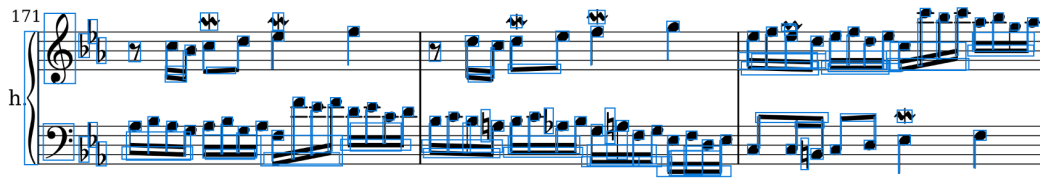
5.3.4 Evaluation and discussion

Table 5.3 presents class-wise average precision (AP) for both baselines. The combination of HRNet and Faster R-CNN appears to have significant potential, achieving very high AP for almost all of the more common classes and representing a new state of the art for music object detection on typeset music. The difference between mAP and AP0.5 is relatively low, which leads to the conjecture that the bounding box predictions are very accurate in terms of position and size. This can be visually confirmed by observing some Faster R-CNN detections as shown in Figure 5.4. Notably missing from the detections are stems, despite being the most common class of symbols. Further analysis is needed as to why these symbols are not properly detected.

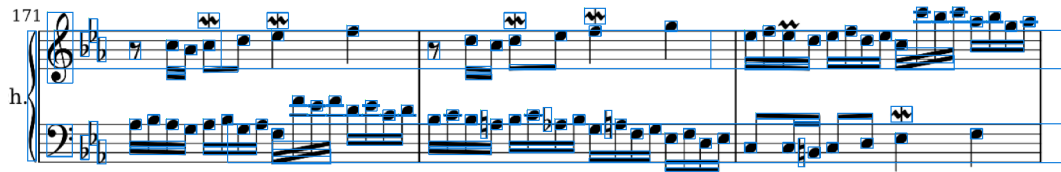
The DWD consistently achieves lower average precision than Faster R-CNN except for the rarest symbols. It also has a bigger spread between mAP and AP0.5. A visual inspection of its detections in Figure 5.4(a) shows that it also finds all of the symbols but often with loose-fitting bounding boxes. Notably, the DWD

detects the stems with a bounding box quality that is very usable in a practical setting, but too loose-fitting to impact the academic metric of AP0.5, let alone mAP. On the other hand, it is clear that DWD struggles considerably with the detection of beams, especially when they are at an angle.

These results show that both systems have their strengths and weaknesses. Currently, none is superior, although Faster R-CNN has made a big leap in performance thanks to the use of HRNet. The problems occurring with the beams further enforce the need for oriented bounding box annotations.



(a) Detections by DWD: every symbol (except for staves and ledgers which are disabled) is detected, although not always with a very accurate bounding box. DWD struggles with beams, sometimes producing multiple or very inaccurate detections.



(b) Detections by Faster R-CNN: all the stems are missed while other symbols are quite accurately detected.

Figure 5.4. Example detection results of the two provided baseline models from the DeepScoresV2 dataset. Both are excerpts from full page detections, cropped for readability.

5.4 The RealScores data

So far, no real-world test data is available to benchmark models on. Such data is crucial to observe how well our models will perform when facing a domain gap. To create a benchmark dataset for real-world OMR, we sourced digitised music scores from the International Music Score Library Project (*IMSLP*) / Petrucci Music Library³. Of the downloaded music scores, only those with specific characteristics were considered for the new test set. The sheets had to be scans or photographs of

³Petrucci Music Library (*IMSLP*): https://imslp.org/wiki/Main_Page

Class	No. Occ	DWD		Faster RCNN		Class	No. Occ	DWD		Faster RCNN	
		mAP	AP0.5	mAP	AP0.5			mAP	AP0.5	mAP	AP0.5
stem	65,088	0.000	0.003	0.004	0.013	keyboardPedalUp	144	0.049	0.180	0.490	0.571
noteheadBlackOnLine	34,785	0.502	0.880	0.934	0.973	rest32nd	140	0.483	0.965	0.992	0.993
noteheadBlackInSpace	33,923	0.489	0.872	0.933	0.969	fingering0	140	0.150	0.646	0.837	0.957
legerLine	23,809	0.000	0.000	0.656	0.854	fingering2	138	0.264	0.723	0.866	0.962
beam	18,846	0.030	0.114	0.819	0.919	fingering4	131	0.300	0.797	0.857	0.962
augmentationDot	5,525	0.035	0.151	0.765	0.871	dynamicS	127	0.043	0.212	0.813	0.945
staff	3,864	0.000	0.000	0.222	0.578	timeSig2	126	0.262	0.772	0.899	0.989
keySharp	3,478	0.448	0.942	0.882	0.967	timeSig1	116	0.349	0.830	0.906	0.997
keyFlat	3,188	0.443	0.921	0.881	0.946	clefCTenor	104	0.523	0.850	0.921	0.959
noteheadHalfOnLine	2,877	0.541	0.930	0.890	0.944	restWhole	94	0.175	0.698	0.069	0.085
noteheadHalfInSpace	2,810	0.510	0.907	0.852	0.913	keyboardPedalPed	93	0.029	0.097	0.563	0.706
tie	2,532	0.007	0.046	0.698	0.859	rest64th	93	0.286	0.669	0.983	0.989
rest8th	2,491	0.441	0.940	0.931	0.988	articStaccatissimoBelow	89	0.034	0.139	0.503	0.949
slur	2,430	0.042	0.159	0.771	0.881	rest128th	88	0.140	0.355	0.952	0.978
flag8thDown	2,281	0.442	0.895	0.926	0.986	articMarcatoAbove	88	0.127	0.545	0.390	0.509
clefG	2,203	0.430	0.880	0.927	0.992	fermataBelow	83	0.322	0.707	0.748	0.945
accidentalSharp	2,133	0.461	0.901	0.940	0.992	timeSig0	83	0.072	0.423	0.862	0.936
restQuarter	2,097	0.382	0.832	0.852	0.976	articTenutoAbove	82	0.007	0.050	0.410	0.685
accidentalNatural	1,941	0.318	0.826	0.900	0.984	ornamentMordent	81	0.286	0.762	0.931	0.988
flag8thUp	1,941	0.301	0.681	0.912	0.989	accidentalDoubleSharp	80	0.181	0.724	0.874	0.963
clefF	1,488	0.470	0.910	0.945	0.982	stringsUpBow	79	0.334	0.676	0.924	1.000
dynamicF	1,437	0.295	0.750	0.803	0.885	restDoubleWhole	77	0.136	0.509	0.896	0.972
timeSig4	1,349	0.361	0.696	0.653	0.723	ornamentTurn	71	0.110	0.577	0.961	1.000
articStaccatoAbove	1,193	0.061	0.250	0.745	0.891	arpeggiato	71	0.001	0.007	0.486	0.741
accidentalFlat	1,164	0.427	0.804	0.899	0.980	articMarcatoBelow	70	0.144	0.655	0.618	0.762
dynamicP	1,096	0.425	0.805	0.786	0.860	dynamicZ	70	0.332	0.974	0.906	0.991
noteheadWholeInSpace	1,008	0.306	0.808	0.868	0.911	timeSig9	69	0.100	0.459	0.908	1.000
repeatDot	876	0.017	0.067	0.833	0.989	stringsDownBow	66	0.548	0.962	0.966	1.000
noteheadWholeOnLine	865	0.387	0.919	0.890	0.939	clef15	63	0.015	0.088	0.627	0.839
rest16th	743	0.544	0.897	0.941	0.988	articStaccatissimoAbove	59	0.049	0.322	0.493	0.955
brace	725	0.000	0.000	0.869	0.969	noteheadDoubleWholeOnLine	57	0.052	0.351	0.372	0.650
restHalf	677	0.149	0.786	0.837	0.955	segno	55	0.471	0.945	0.969	1.000
dynamicM	533	0.292	0.782	0.698	0.807	ornamentTrill	52	0.420	0.943	0.856	0.997
articAccentAbove	521	0.369	0.871	0.818	0.960	flag32ndUp	49	0.231	0.674	0.502	0.810
articStaccatoBelow	503	0.017	0.078	0.641	0.790	coda	49	0.146	0.288	0.963	0.980
timeSig3	401	0.124	0.440	0.419	0.470	flag128thUp	45	0.035	0.185	0.947	0.999
flag16thDown	335	0.222	0.551	0.910	0.970	flag128thDown	42	0.030	0.288	0.948	1.000
tuplet3	329	0.092	0.362	0.765	0.941	flag64thDown	42	0.216	0.621	0.887	0.923
timeSig8	322	0.257	0.657	0.682	0.852	timeSig7	40	0.222	0.801	0.885	0.995
dynamicCrescendoHairpin	298	0.116	0.237	0.807	0.953	flag64thUp	29	0.028	0.095	0.802	0.850
articAccentBelow	274	0.398	0.864	0.776	0.963	articTenutoBelow	27	0.000	0.000	0.000	0.000
flag16thUp	263	0.370	0.813	0.937	1.000	restHBar	27	0.040	0.213	0.000	0.000
clefCAlto	255	0.396	0.649	0.903	0.970	ottavaBracket	26	0.000	0.000	0.173	0.300
flag32ndDown	239	0.010	0.017	0.000	0.000	tupletBracket	25	0.000	0.000	0.468	0.684
clef8	230	0.156	0.485	0.584	0.691	noteheadDoubleWholeInSpace	21	0.040	0.194	0.000	0.000
fingering1	226	0.081	0.307	0.860	0.959	ornamentTurnInverted	17	0.321	0.795	0.961	0.994
tuplet6	207	0.053	0.295	0.893	0.977	tuplet5	4	0.055	0.250	0.000	0.000
dynamicDiminuendoHairpin	192	0.053	0.153	0.747	0.918	dynamicR	4	0.088	0.125	0.000	0.000
timeSig6	185	0.197	0.794	0.461	0.574	fingering5	3	0.115	0.136	0.783	0.917
fermataAbove	184	0.227	0.741	0.846	0.966	tuplet1	0	0.000	0.000	0.000	0.000
keyNatural	183	0.265	0.721	0.867	0.993	tuplet8	0	0.000	0.000	0.000	0.000
timeSig5	146	0.007	0.044	0.009	0.014	accidentalDoubleFlat	0	0.000	0.000	0.000	0.000
caesura	146	0.041	0.204	0.757	0.916						
fingering3	146	0.137	0.443	0.852	0.947						
mean								0.203	0.503	0.700	0.799
weighted mean								0.219	0.422	0.608	0.676

Table 5.3. Classwise Average Precision at 0.5 overlap (AP0.5) as well as Mean Average Precision (mAP) of DWD and Faster R-CNN.

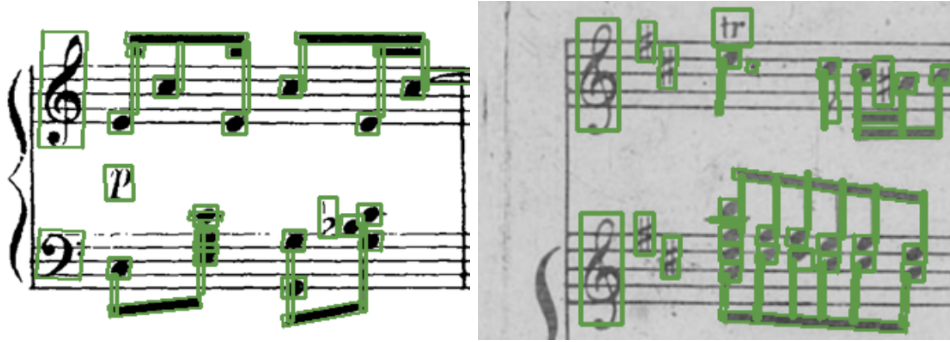


Figure 5.5. Example snippets from two RealScores pages with ground truth annotations overlaid.

music scores and be visibly non-synthetic, meaning that they come with scanning artefacts, discolourations, stains, folds, be angled, and have other imperfections. Music scores that were handwritten, of very bad quality, or using non-standard notation were considered out of scope for this work. The selected samples had to be annotated by hand using ScorePad’s current OMR pipeline [Stadelmann et al., 2018]. The resulting test set consists of 12 music sheets with a total of 12 553 annotations that we name *RealScores*. The annotations are stored in the same format that *DeepScoresV2* [Tugener et al., 2021] introduced. Due to its limited size, only 61 of the original 136 classes are present. Excerpts from two samples with their corresponding annotations are visible in Figure 5.5.

In a second step, we sourced a number of “blank” pages from the aforementioned Petrucci Music Library. This was possible because many uploaded music scores would be sourced from completely scanned books, including the front and back covers. Such scans sometimes contain blank pages without any written music, but all the perturbations that naturally occur on sheets of paper. This is a valuable source of real-world noise that can be overlaid with synthetic data. We manually looked through the sourced data for suitable blanks, then converted them into pictures and normalised their size to fit with the synthetic data of *DeepScoresV2*. A total of 51 such blank pages were selected – 30 of which have a significant portion of the sheet border visible, and 21 do not. Figure 5.6 shows six of those pages.

5.5 ScoreAug

We propose a sophisticated data augmentation scheme to address the domain gap that we call *ScoreAug*. With *ScoreAug*, input samples first can be blurred, get salt-



Figure 5.6. Example blank pages.

and-pepper-like noise, get irregular edges in the border area, be rotated by a small angle, or become augmented with other irregularities not found in a synthetic dataset like *DeepScoresV2*. Additionally, we go one step beyond these algorithmic perturbations and complement them by overlaying them on our blank pages from the *RealScores* dataset. Using this combination of augmentation techniques, we aim to bring synthetic data close enough to the real-world domain to train models that generalise to real-world inputs. For a given synthetic input image, we select one out of our set of the 51 blank pages. To increase variability, the blank page and the synthetic data undergo a variety of further augmentations, as shown in Table 5.4.

To ensure alignment with the transformed image data, the ground-truth bounding boxes also undergo the same transformations. Upon completing these augmentations, the foreground and background are merged by preserving the darker pixel at each position. This means that darker pixels overpower the lighter shades, preserving the dark symbols from the augmented synthetic dataset (the foreground) and replacing the pixels of its white background with the darker

	Blanks	Scores
Salt and Pepper Noise	-	P_{snp}
No additional Augmentations	-	P_{aug}
Horizontal Flip	50%	-
Vertical Flip	50%	-
Crop and Resize	20%	-
Randomise Brightness	50%	-
Higher Contrast	-	20%
Small Angle Rotation	60%	60%
Additional Brightness	-	40%
Gaussian Blur	-	P_{blur}

Table 5.4. Probabilities of augmentations as part of ScoreAug that can be applied to either the blanks, synthetic scores, or both at the same time. Note that P_{aug} decides how likely any other augmentations (after the salt and pepper noise) will be applied, in order to not only feed ScoreAugmented samples to the model. Our final model uses $P_{\text{snp}} = 0\%$, $P_{\text{aug}} = 30\%$, $P_{\text{blur}} = 10\%$.

pixels from the augmented blank pages (the background). This yields optically similar results to real-world scanned music scores as seen in Figure 5.7, which can be adapted with hyperparameters (P_{snp} , P_{aug} , P_{blur}) to adjust to one’s needs.

Experimental Setup: To measure the impact of *ScoreAug*, we trained one baseline model with *ScoreAug* and another without it – each for 2 000 epochs. Both models were trained on half-resolution, cropped samples to allow for larger batch sizes and faster convergence. During training, we used a learning rate of $\alpha = 2.5 \cdot 10^{-3}$ throughout and used linear warmup with a ratio of $\frac{1}{3}$ for the first 500 epochs. We

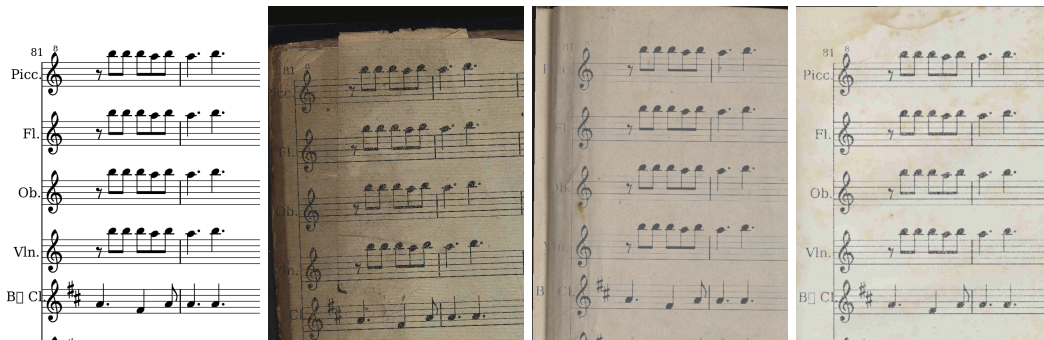


Figure 5.7. ScoreAug examples (top right, bottom row) derived from the same synthetic sample (top left).

<i>DeepScoresV2 dataset</i>	
Model	AP (overlap = 0.25)
Baseline	87.6%
<i>ScoreAug</i>	86.0%
<i>ScoreAug</i> + <i>Finalise</i>	83.3%

<i>RealScores dataset</i>	
Model	AP (overlap = 0.25)
Baseline	36.0%
<i>ScoreAug</i>	56.5%
<i>ScoreAug</i> + <i>Finalise</i>	73.7%

Table 5.5. The AP for the baseline model and models with *ScoreAug* and *Finalise* data augmentation on the *DeepScoresV2* and the *RealScores* datasets.

observed that the models lack global awareness (e.g. predicting noteheads at the corner of the page), therefore we trained some models an additional 200 epochs on full pages, we denote this step as *Finalise*. During evaluation, we make sure to only consider the results of classes that have at least one positive prediction per model. We evaluate our models using average precision (AP) at an overlap of 25%. We use this unusually low overlap threshold due to the very small object sizes common in MOR, which cause detections that are very usable in practice to often be below the 50% mark.

Results: Thanks to *ScoreAug* and *Finalise* we observe an absolute increase in AP of roughly 40% compared to models trained for the same number of epochs and without using both (see Table 5.5). We observe that on the source Dataset *DeepScoresV2* the performance slightly degrades from 87.6% to 83.3%. However, this is to be expected since we move from a model specifically trained on and for synthetic data to one that can handle a much wider variety of data.

5.6 Conclusion and future work

First, we presented *DeepScoresV2*, an enhanced version of the *DeepScores* dataset for music object detection. *DeepScoresV2* has a wider range of annotated symbols as well as oriented bounding boxes for more accurate and semantically informative detections. The presented baselines show that current models already perform quite well on *DeepScoresV2*, achieving a new state of the art, especially with a Faster R-CNN detector using an HRNet backbone. However, additional work is

needed regarding small objects such as stems as well as rare objects. The newly provided ground truth for oriented bounding boxes can serve to develop new models that increase prediction accuracy on rotated objects with a non-uniform aspect ratio.

Second, we introduced ScoreAug to bridge the domain gap between synthetic and realistic data and created RealScores to test how well a trained model generalizes to real-world data. Using ScoreAug + Finalise we were able to more than double the detection performance of our model on the realistic data. This shows the usefulness of our setup of combining real perturbations with synthetic data and proves that data augmentation is a worthwhile approach for addressing domain shifts.

Evaluation metrics are designed to generate an accurate description of the performance of a model by a few representative numbers. There is a huge disparity between the metrics for the stem detections of DWD and how we judge the same detections of the stems visually (seen in Figure 5.4). This leads to the insight that AP0.5 and mAP, which have been designed for general object detection and only consider detections with an overlap of at least 50% between predicted and ground truth bounding boxes, do not fulfil this goal in every case, especially not for very small objects. Therefore, we conclude that AP0.5 and mAP are not well-suited to judge music object detection systems, and the field should strive to find or develop a more appropriate metric.

We also encourage OMR researchers to use the newly available staff and rhythm information to build more powerful models that can directly infer higher-order semantic information.

Chapter 6

Automated machine learning under constrained resources

A main driver behind the digitization of industry and society is the belief that data-driven model building and decision making can contribute to higher degrees of automation and more informed decisions. Building such models from data often involves the application of some form of machine learning. Thus, there is an ever growing demand in work force with the necessary skill set to do so. This demand has given rise to a new research topic concerned with fitting machine learning models fully automatically—AutoML. This chapter gives an overview of the state of the art in AutoML with a focus on practical applicability, introduces portfolio hyperband a novel AutoML setup, and presents an investigation on CNN architectures for automated deep learning which yields a useful design pattern: the sponge effect.

6.1 Introduction

Organisations, private and public, have understood that data analysis is a powerful tool to learn insights on how to improve their business model, decision-making, and even products [Braschler et al., 2019]. A central step in the analysis process is often the construction and training of a machine learning model, which itself entails several challenging steps, most notably feature preprocessing, algorithm selection, hyperparameter tuning and ensemble building. Usually, a lot of expert

This chapter is based on excerpts of [Tuggener et al., 2019], previously published at SDS 19 as a conference paper and excerpts of [Tuggener et al., 2020], previously published in AI as a journal paper. This chapter focuses on the contributions to the abovementioned publications by the author of this thesis.

knowledge is necessary to successfully perform all these steps [Meier et al., 2018]. The field of automated machine learning (AutoML) aims to develop methods that build suitable machine learning models without (or as little as possible) human intervention [Hutter et al., 2019].

In this chapter, we first give our insights into why AutoML is currently heavily researched and also of practical relevance in business and industry (see Section 6.2). Section 6.3 introduces core concepts of AutoML and surveys the current state of AutoML. In Section 6.4, we present portfolio hyperband and benchmark it against the scientific state of the art and an industrial prototype. Lastly in Section 6.5 we evaluate design choices for automated deep learning on vision data and extract a useful design pattern: the sponge effect.

6.2 Impact of practical machine learning

In recent years, machine learning has been applied to more and more domains. Industrial applications for example, such as predictive maintenance [Susto et al., 2015; Li et al., 2014] and defect detection [Figueiredo et al., 2011], enable companies to be more proactive and improve efficiency. In the area of healthcare, patient data have helped addressing complex diseases, such as multiple sclerosis, and support doctors in identifying the most appropriate therapy [Stühler et al., submitted]. In the insurance and banking sectors, risks in loan applications [Handzic et al., 2003] and claims processing [Viaene et al., 2005; Pérez et al., 2005] can be estimated, enabling automated identification of fraudulent patterns. Finally, advances in sales and revenue forecasting support supply chain optimization [Tsoumakas, 2018].

However, the process towards building such actionable machine learning models, able to generate added value to the business, is time-consuming and error-prone, if done manually; performance of different models should be compared, considering different algorithms, hyperparameter tuning and feature selection. This process is highly iterative and, as such, is an ideal candidate for automation. With the use of AutoML, the data scientist is freed from this tedious task and can focus on more creative tasks, delivering more value to the company. New business cases can be identified, assessed and validated in a rapid-prototype-fashion.

In practice, AutoML can provide different kind of insights. Already at an early stage, running different models on the input data can provide feedback on how suitable the data is for predicting the given target. When multiple models built from a wide spectrum of algorithms do not perform significantly better than the baseline, this can be seen as an indication of insufficient predictive power in the

data. However, good models will be generated and the data scientist is left with the choice of deploying the best generated model or to create an ensemble from a selection of models. Finally, there is a by-product in AutoML when optimizing over the feature set as well: One can derive an estimate of feature importance by statistical analysis of the model quality depending on which features are used as input to the models.

Thus, the introduction of AutoML tools in a company can drastically increase efficiency of the work of a data scientist. Taking the example of one of our predictive maintenance projects in the area of public transport, where a team of three data scientists worked full time for weeks, a machine learning model with an out-of-sample area under the curve (AUC) of 0.81 was developed. A few months after that milestone, the prototype of the AutoML tool described later in Section 6.4 (DSM), using the same dataset (and no other help or information) was used as a benchmark to evaluate the benefits of this tool. It resulted in an automatically generated model that was slightly out-performing the manually engineered model with an AUC of 0.82 after a run time of half an hour.

6.3 Survey of the current state of automated machine learning

Automated machine learning (AutoML) and by extension, AutoDL is usually framed as the *combined algorithm selection and hyperparameter optimization (CASH) problem* [Thornton et al., 2013], which encompasses choice and parameterization of both a machine-learning model and a respective training algorithm - all without any human input. Ultimately, a full solution finds the best machine learning pipeline for raw (un-preprocessed) feature vectors in the shortest time for a given amount of computational resources. A complete pipeline includes data cleaning, feature engineering (selection and construction), model selection, hyperparameter optimization and finally building an ensemble of the top trained models to obtain good performance on unseen test data. Optimizing the entire machine learning pipeline (that is not necessarily differentiable end-to-end) is a challenging task, hence diverse approaches have been investigated. This section comprehensively surveys the current state of AutoML.

Feature engineering: Feature preprocessing, representation learning and selecting the most discriminant features for a given classification or regression task are problems targeted by the literature. Gaudel et al. [Gaudel and Sebag, 2010] consider feature engineering as a one-player game and train a reinforcement

learning-based agent to select the best features. To do so, they first model the feature selection problem as a Markov Decision Process (MDP). Second, they propose a reward associated with generalization error in the final status. The agent learns an optimal policy to minimize the final generalization error.

Explorekit [Katz et al., 2016] not only iteratively selects the features but also generates new feature candidates to obtain the most discriminant ones. Katz et al. use normalization and discrimination operators on a single feature to generate unary features. They additionally combine two or more features to generate new candidates and train a feature rank estimator based on meta-features extracted from the datasets and candidates. The feature with the highest rank that increases the classification accuracy above a certain threshold is added to the selected feature set in every iteration.

To reduce the computational complexity of iterative feature selection methods, *Learning Feature Engineering* (LFE) [Nargesian et al., 2017] learns the effectiveness of a transformation based on previous experiments. The original feature space is subsequently mapped via the optimal transformation to compute a discriminant feature representation.

AutoLearn [Kaul et al., 2017] is a regression-based algorithm for automatic feature selection. The proposed algorithm starts by filtering the original features and discard the ones with small information gain. Subsequently, feature pairs are filtered based on distance correlation to omit dependent pairs. The new features are generated based on the remaining pairs using ridge regression. Ultimately, the best features are the ones with the highest information gain and stability [Meinshausen and Bühlmann, 2010]. *AutoLearn* has been applied to various datasets, including gene expression data.

Meta-learning here refers to methods that try to leverage meta information about the problem at hand, e.g. the dataset as well as the available algorithms and their configurations, to improve the performance of an AutoML system. This meta-information is often gathered and processed using machine learning methods, thus in a sense applying the discipline to itself. The meta information about datasets often consists of some basic statistical reference numbers and some landmarks, i.e. performance figures of simple algorithms [Pfahringer et al., 2000].

Learning curve prediction attempts to learn a model that predicts how much the performance of a learner will improve if given more training time [Klein et al., 2016]. Another take on this idea are attempts to predict the running time of algorithms [Eggersperger et al., 2018]. Instead of predicting absolute performance figures, it has sometimes proven beneficial to predict a ranking of the available algorithms to choose from [Brazdil and Soares, 2000].

Meta-learners in the context of neural networks aim to improve the optimizer

of a deep or shallow (convolutional) neural network (CNN) to reach a minimum as quick as possible through automatic hyper-parameter tuning. Andrychowicz et al. [Andrychowicz et al., 2016] learn to predict the best set of hyperparameters for optimizing neural networks with gradients and a Long Short-Term Memory [Hochreiter and Schmidhuber, 1997] network. Similarly, Chen et al. [Chen et al., 2017] train an optimizer for simple synthetic functions such as Gaussian Processes. They demonstrate that the optimizer generalizes to a wide range of black-box problems. For instance, the trained optimizer is used to tune the hyperparameters of a Support Vector Machine [Cortes and Vapnik, 1995] without accessing the gradients of the loss function with respect to the hyperparameters.

Architecture search literature discusses methods for finding the best performing architecture for neural networks automatically without human expert intervention. Early work focused on reducing the search space of architectures e.g. by graph rewriting [Kitano, 1990]. A more general approach aimed to compactly encode weights of large networks in an universal programming language [Schmidhuber, 1997]. Elsken et al. [Elsken et al., 2018] propose *Neural Architecture Search by Hill-climbing* (NASH) using local search. The algorithm starts with a well-performing (preferably pretrained) convolutional architecture (parent). Then, two types of network morphisms (transformations) are randomly applied to generate deeper or wider architecture children from the original parent network. The children architectures are trained, and the best-performing architecture qualifies for the next round. The algorithm iterates until the validation accuracy saturates.

Real et al. [Real et al., 2017] propose an evolutionary architecture search based on a pairwise comparison within the population: the algorithm starts with an initial population as parents, and every network undergoes random mutations such as adding and removing convolutional layers and skip connections to produce offspring. Subsequently, parent and child compete in pairwise comparison, with the winner model staying in the population and the loser being discarded.

In contrast to evolutionary algorithms, where larger and more accurate architectures are desired, He et al. [He et al., 2018] automatically search for compressing a given CNN for mobile and embedded applications. Their *AutoML for Model Compression* (AMC) algorithm trains a reinforcement learning agent to estimate the sparsity ratio of each layer and then compress the layers sequentially.

Zoph et al. [Zoph and Le, 2017] train a controller using reinforcement learning and a Recurrent Neural Network (RNN) to tune the hyperparameters of a deep CNN architecture such as width and height of filters and strides. The RNN controller is trained using a policy gradient method to maximize the network's accuracy on a hold-out set of data.

Hyperparameter optimization is a crucial step for solving the entire CASH problem, with *Bayesian optimization* [Brochu et al., 2010] being the most prominent specimen of respective approaches. The goal here is to build a model of expected loss and variance for every input. After each optimization step, the model (or current belief) is updated using the a posteriori information (hence the name Bayesian).

An acquisition function is defined that decides at which location to sample the next true loss, trading off regions of low expected loss (exploitation) and regions of high variance (exploration). Usually, Gaussian Processes are the model of choice in Bayesian optimization; alternatively, Random Forests have been used to model the loss surface of the hyperparameters as a Gaussian distribution in *Sequential Model-based optimization for general Algorithm Configuration* (SMAC) [Hutter et al., 2011] or the *Tree-structured Parzen Estimator* [Feurer et al., 2014].

Model free methods include *Successive Halving* [Jamieson and Talwalkar, 2016] and built on it *Hyperband* [Li et al., 2017], which uses real time optimization progress to narrow down a set of competing hyperparameter configurations over the duration of a full optimization run, possibly with many restarts. A slight variation of this are *Evolutionary Strategies* that also allow for perturbations of the individual configurations during training [Jaderberg et al., 2017]. In the special case where the optimizee as well as the optimizer are differentiable, multiple iterations of the optimizer can be unrolled, and an update for the hyperparameters can be computed by using gradient descent and backpropagation [Maclaurin et al., 2015].

Pipeline Optimization: Complete machine learning pipelines, including feature preprocessing, model selection, hyperparameters optimization and building ensembles, are constructed based on different views of the entire problem. Bayesian optimization, Genetic programming [Banzhaf et al., 1998], and bandit optimization inspired developing various pipeline optimization frameworks.

Auto-sklearn is aimed at solving the CASH problem using meta-learning, Bayesian optimization and ensemble building. First, it extracts meta-features of a new dataset such as task type (classification or regression), number of classes, the dimensionality of the feature vectors, number of samples and so on. The meta-learner of Auto-sklearn uses these meta-features to initialize the optimization step based on previous experience on similar data sets (similar according to the meta features). Then, preprocessing and model hyperparameters are iteratively enhanced using Bayesian optimization. Ultimately, a robust classifier or regression model is built based on an ensemble of models trained during the iterative optimization.

Tree-based Pipeline Optimization Tool (TPOT) is an algorithm uses genetic

programming to optimize the machine learning pipeline. It searches for the best pipeline including feature processing, model and hyperparameters for a given classification or regression task. The feature processing module of TPOT works in conjunction with feature selection and generation. The feature generation block performs the kernel trick [Schölkopf, 2001] or dimensionality reduction methods. The optimization is done using genetic programming: first, the algorithm generates some tree-based pipelines randomly. Then, it selects the top 20% of the generated population based on cross-validation accuracy, and produces 5 descendants from each by randomly changing a point in the pipeline. The algorithm continues until a stopping criterion is met.

6.4 Portfolio Hyperband

In this section, we evaluate the usefulness of classical AutoML for application in business and industry by empirically comparing the most successful automated machine learning algorithms with (a) an industrial prototype as well as (b) a straight-forward improvement inspired by Hyperband [Li et al., 2017; Feurer et al., 2018] (c). This selection spans a wide range of different approaches for pipeline optimization (see Section 6.3) to tackle the CASH problem: the industrial prototype *DSM* [Stadelmann et al., 2018] uses random model and hyperparameter search and thus serves as a baseline; Auto-sklearn [Feurer et al., 2015] has won the recent AutoML challenges [Guyon et al., 2017]. Additionally, we report results with TPOT [Olson et al., 2016], which is developed based on genetic programming [Banzhaf et al., 1998] instead of Auto-sklearn’s Bayesian optimization. We give a brief overview of each system below:

Data Science Machine (DSM): The Data Science Machine (DSM) [Stadelmann et al., 2018] has been developed for both in-house and client-related data science projects by PwC. DSM includes a portfolio of open-source machine learning algorithms, and also offers the possibility to add custom algorithms through a language-agnostic API. Given a dataset, the tool automatically optimizes over the set of algorithms, features, and hyperparameters. The developed solution offers multiple optimization strategies, e.g. tuneable genetic algorithms. In the context of this chapter, however, we limited it to use random sampling to serve as a baseline.

Auto-sklearn [Feurer et al., 2015]: We used the algorithm explained the Section above to find the best pipeline within the time-budget computed for training 100 models by DSM. Auto-sklearn is slow in start [Feurer et al., 2015]; however, it eventually reaches a solution very close to the optimum. The method

benefits from meta-learning using similar datasets and it is usually pretrained on OpenML datasets [Vanschoren et al., 2013] in the challenges. We applied the base model of Auto-sklearn to compare the exploration efficiency of the different approaches.

TPOT [Olson et al., 2016]: This algorithm uses tree-based classifiers which is similar to the second entry of the latest AutoML challenge [Guyon et al., 2017]. TPOT differs from the other presented methods since it used Genetic programming for optimization. We initialized the algorithm with a population of 20 tree-based pipelines and stopped the optimization with the same time budget as DSM and Auto-sklearn.

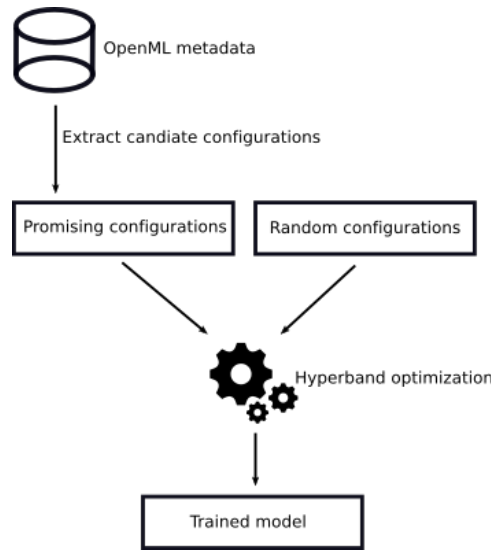


Figure 6.1. Schematic overview of the Portfolio Hyperband workflow.

Portfolio Hyperband [Feurer et al., 2018; Li et al., 2017]: Inspired by PoSH Auto-sklearn [Feurer et al., 2018] that combines a portfolio of initial configurations with successive halving (SH) and Bayesian optimization, we built a system that combines a portfolio with Hyperband [Li et al., 2017]. Our goal was to combine the portfolio variant of meta-learning, which is very simple and fast, with Hyperband which should give the system a good asymptotic performance. At the basis of Hyperband is the successive halving algorithm that starts with an initial set of configurations and trains all of them for a fixed amount of time; then, the less performing half of the configurations is dropped. This process is repeated until only the best performing configuration remains, hence the name successive halving. An issue with SH is that it is unclear with how many initial configurations to start the process. Hyperband performs a geometric search to explore the

Dataset	Task	Metric	DSM		Auto-Sklearn		TPOT		Portfolio Hyperband	
			Test	Time	Test	Time	Test	Time	Test, full time	Test, 10 Min
Cadata	Regression	R2 (coefficient of determination)	0.7119	55.0	0.7327	54.9	0.7989	54.6	-	-
Christine	Binary classification	Balanced accuracy score	0.7146	99.4	0.7392	99.3	0.7442	105.1	0.753	0.744
Digits	Multiclass classification	Balanced accuracy score	0.8751	201.2	0.9542	201.2	0.9476	207.2	-	-
Fabert	Multiclass classification	Accuracy score	0.8665	77.5	0.8908	77.4	0.8835	78.5	-	-
Helena	Multiclass classification	Balanced accuracy score	0.2103	190.2	0.3235	216.4	0.3470	197.5	-	-
Jasmine	Binary classification	Balanced accuracy score	0.8371	24.1	0.8214	24.0	0.8326	25.9	-	-
Madeline	Binary classification	Balanced accuracy score	0.7686	48.3	0.8896	48.2	0.8684	53.0	0.868	0.848
Philippine	Binary classification	Balanced accuracy score	0.7406	56.3	0.7634	56.2	0.7703	56.4	0.741	0.753
Sylvine	Binary classification	Balanced accuracy score	0.9233	28.9	0.9350	28.9	0.9415	29.0	0.947	0.916
Volkert	Multiclass classification	Accuracy score	0.8154	122.3	0.8880	122.2	0.8720	125.5	-	-
Average Performance			0.7463	90.31	0.7938	92.85	0.8006	93.26		

Table 6.1. Performance of selected automated machine learning algorithms on AutoML challenge data sets [Guyon et al., 2017]. "Test" refers to our own test split; "Time" is in minutes; "Portfolio Hyperband" only tested on binary classification.

trade-off between individual training time and the amount of different initial configurations.

To create a collection of promising configuration candidates, we surveyed all the meta data available on OpenML [Vanschoren et al., 2013] and extracted a list of different configurations that worked well for binary classification. This list is used to seed Hyperband. Every run also uses some random initialisations to improve long term performance. To test the viability of Portfolio Hyperband we applied it to binary classification only.

To compare the presented automated machine learning approaches, we select some datasets with various classification and regression tasks from previous AutoML challenges [Guyon et al., 2017]. We chose all datasets that are suitable TPOT and Auto-sklearn. Because the official test labels are not public we used our own training, validation and test split. For the DSM (random search as a baseline), we randomly pick a set of 100 models and hyperparameters, train the models and extract the best-performing ones for each data set. During the experiments, the time budget used for DSM is computed and used as the reference value for the rest of the optimization methods. Auto-sklearn, TPOT and Portfolio Hyperband subsequently use this time budget to find the best pipeline. We start training from scratch without pretraining on any similar data set; therefore, the meta-learning block of Auto-sklearn is not well tuned. Since Portfolio Hyperband turned out to find good initial models very fast, we also report its performance after a considerably shortened period of 10 minutes.

The results of the benchmarking are presented in Table 6.1. Numerical evaluation suggests that all three sophisticated approaches outperform the DSM in baseline mode (only random search) consistently, but not by a large margin.

However, the accuracy of the three approaches from current research is quite similar, while Portfolio Hyperband appears to be especially quick.

6.4.1 Intermediate conclusions

None of the presented methods is superior to all others. However, we can identify two major takeaways. First, we note that random search (DSM) is still quite competitive, especially when constrained to a relatively small set of tried and true options. It falls short to systems that leverage meta-learning, especially with very constrained time budgets. A take-away from this is that sometimes it can make sense to invest in additional and faster hardware for parallel search rather than in a very sophisticated AutoML solution.

Second, we find that the use of meta data to guide the search or even pre-trained models is one of the most potent ways to speed up AutoML. Working with completely unrelated data (e.g. from OpenML) already yields a sizeable speed-up in the Portfolio Hyperband system. The closer the data on which the meta-learner is trained (offline data) and the data to be analyzed (online data) are in distribution, the stronger this effect gets - up to the point where the model can be almost fully trained on the offline data and then is only fine tuned using the online data. Meta learning therefore is especially attractive for business cases where a continuous data generating process (e.g. monthly reports, continuous sensor feedback) produces new data that is similar in distribution to the old data already seen from the same source.

6.5 AutoDL for Vision and the sponge effect

This section presents our systematic evaluation of automated deep learning for vision methods we conducted in the context of the 2019-2020 ChaLearn AutoDL challenge [Liu, Guyon, Junior, Madadi, Escalera, Pavao, Escalante, Tu, Xu and Treguer, 2019] and the sponge effect, a phenomenon we discovered thereby. The AutoDL competition comprised several sub-challenges focused on different data modalities such as images, video, audio, text, and tabular data. The challenge provides multiple curated datasets for each modality, published a code framework, and hosted online evaluation servers [Liu, 2018]. They defined their own novel evaluation metric Area under the Learning Curve (ALC) [Liu, Guyon, Junior, Madadi, Escalera, Pavao, Escalante, Tu, Xu and Treguer, 2019], which is focused on judging any-time performance and thus encouraging fast practically viable solutions.

The goal of the vision AutoDL sub-challenge is the development of a system that can perform multi-label and multi-class classification of images and videos of any type without human intervention. Seven datasets are available for offline model development and training (for details see Table 6.2). Developed models can then be validated on five unknown datasets (two containing images, three containing videos). The final evaluation is performed on five private datasets.

Based on the results of our evaluation we establish specific patterns for designing successful vision systems for the metrics defined by the AutoDL challenge [Liu, Guyon, Junior, Madadi, Escalera, Pavao, Escalante, Tu, Xu and Treguer, 2019]. As a baseline model we adopt MobileNetV2 [Sandler et al., 2018], inspired by the runner-up of AutoCV1 [Lee, 2019]. Initial tests showed that pretraining on ImageNet [Russakovsky et al., 2015] is necessary, therefore all subsequent experiments are conducted using pre-trained models.

Table 6.2. Vision: Summary of vision datasets.

Dataset	Content	Modality	Size	No. Training Samples	No. Classes
Chucky	Everyday Objects	Images	$32 \times 32 \times 1$	48,061	100
Decal	Satellite Images	Images	Varying	634	11
Hammer	Medical Images	Images	$400 \times 300 \times 3$	8050	7
Pedro	Pedestrian Images	Images	Varying	80,095	26
Katze	Common Actions	Videos	$120 \times 160 \times 3$	1528	6
Kraut	Common Actions	Videos	$120 \times 160 \times 3$	1528	4
Kreatur	Common Actions	Videos	$60 \times 80 \times 3$	1528	6

6.5.1 Base architecture selection

The basic neural network architecture is a crucial choice for any deep-learning system. This is especially true for automated deep learning where next to predictive performance, flexibility is paramount. The recently introduced EfficientNets [Tan and Le, 2019] are a strong candidate architecture as they combine top performance with lean design. The most appealing feature of the EfficientNet in the context of AutoDL is that the architecture incorporates two natural scaling parameters, which can be used to scale the base network up (wider and deeper) or down (shallower and narrower). To test our hypothesis that EfficientNets are a solid choice in an AutoDL setting, we benchmark the smallest configuration used in the original publication, EfficientNet-B0 (scaling: 1, 1), as well as an even smaller version we call EfficientNet-mini (scaling: 0.8, 0.8), against our

baseline of MobileNetV2 (using otherwise similar hyperparameters). On top of each model, a classifier consisting of two fully connected layers of size 512 and no regularization is used.

Table 6.3. Vision: Numerical evaluation of different network architectures on the image and video training datasets.

	Datatype Dataset	Image					Video	
		Chucky	Decal	Hammer	Pedro	Katze	Kraut	Kreatur
MobileNetV2	Return Time (s)	15.86	20.35	26.42	25.53	21.54	21.29	16.24
	ALC	0.7536	0.7759	0.7562	0.7491	0.8508	0.6551	0.8678
	NAUC	0.6853	0.8568	0.8538	0.8712	0.9487	0.7433	0.9521
	Overfit?	Yes	Minor	No	No	No	No	No
EfficientNet-mini	Return Time (s)	75.65	76.53	85.51	83.48	83.28	82.77	80.61
	ALC	0.5642	0.5804	0.5896	0.5938	0.6559	0.4952	0.6543
	NAUC	0.779	0.7852	0.8371	0.9043	0.9321	0.6992	0.9258
	Overfit?	Yes	Minor	No	No	No	No	No
EfficientNet B0	Return Time (s)	78.87	77.85	88.02	85.57	84.27	85.77	84.2
	ALC	0.5880	0.5831	0.6165	0.6101	0.6542	0.4906	0.6584
	NAUC	0.8233	0.7657	0.90	0.9231	0.9548	0.6905	0.9489
	Overfit?	Yes	Yes	No	No	No	No	No

The benchmark results in Table 6.3 show that EfficientNet-B0 has high predictive performance as indicated by the NAUC (final performance) score. In terms of ALC (anytime performance), the much smaller and faster MobileNetV2 scores highest for each dataset. EfficientNet-mini is not a worthwhile compromise, since it is only marginally faster than EfficientNet-B0 while having a considerably worse predictive performance. These results lead to the verdict that in a time-sensitive setting MobileNetV2 is preferred, while under less tight time constraints or for more complicated vision tasks EfficientNet-B0 is a good choice. For strong anytime performance, it is recommended to use a combination of both architectures and select the more appropriate model based on the availability of data and complexity of the image classification task.

6.5.2 Classifier design

Apart from the base architecture, the size and design of the fully connected classifier layers are the most important design choices when building a Convolutional Neural Network (CNN). The first question that needs to be answered is how the extracted features are fed into the classifier. Two common options are to reshape the output to have dimension one in height and width (flatten), or to compute the average over height and width (spatial pooling). For videos, there is

an additional time dimension that can be pooled (temporal pooling). Experiments on all datasets (see Table 6.4) conclusively show that spatial pooling is the best option.

Table 6.4. Vision: Comparison of different pooling methods for image and video classification, using the MobileNetv2 architecture.

Datatype Dataset Measure	Image						Video					
	Chucky		Decal		Hammer		Pedro		Katze		Kraut	
	ALC	NAUC	ALC	NAUC	ALC	NAUC	ALC	NAUC	ALC	NAUC	ALC	NAUC
Flatten	0.6922	0.5894	0.7105	0.8078	0.6992	0.7752	0.7371	0.8642	0.8311	0.9347	0.6053	0.7179
Spat. pooling	0.7511	0.6737	0.7791	0.8637	0.7478	0.8358	0.7483	0.8951	0.8512	0.9540	0.6646	0.8650
Tmp. pooling	—	—	—	—	—	—	—	—	0.8290	0.9192	0.6246	0.6870
Spat. & tmp.	—	—	—	—	—	—	—	—	0.8445	0.9359	0.6305	0.6430
											0.7531	0.8910
											0.8447	0.9346

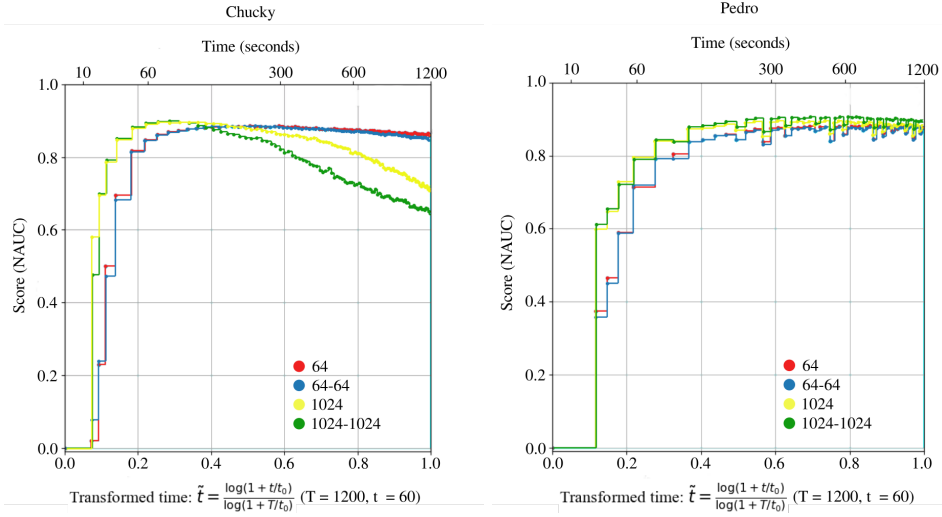


Figure 6.2. Vision: Learning curves as function of training time for MobileNetV2 with varying layout of the fully connected classifier layer(s): 1 layer with 64 neurons (1×64 , red), 2×64 (blue), 1×1024 (yellow) and 2×1024 (green), shown for image datasets Chucky (left) and Pedro (right).

The second important characteristic is the number of layers and number of neurons per layer of the fully connected classifier. We tested different one- and two-layer classifiers from 64 neurons in a single layer up to two times 1024 neurons in two layers. Figure 6.2 shows a very interesting phenomenon: big classifiers are able to absorb information faster and therefore learn much quicker during the first couple of iterations. We dubbed this the *sponge effect*, as the fast information intake resembles a sponge being immersed in water. The *sponge effect* can be

exploited whenever a network design needs to be adapted for fast training speeds. As we observe that this phenomenon is not only relevant to vision, we will present a more detailed, multi-modal, investigation in Section [6.5.3](#).

6.5.3 The sponge effect in multiple modalities

As discussed above, Figure [6.2](#) reveals the interesting phenomenon we dubbed *sponge effect* that large fully connected classifiers initially learn much faster than small ones. In this section, we study this effect more closely for two modalities: vision and audio. We do this by fitting our models with different classifiers - small to large, one or two layers - on all five datasets per modality. Since the datasets and models are much smaller for audio than for vision, we use 16, 32, 64 and 128 neurons per layer for audio and 64, 256, 512 and 1024 neurons per layer for vision.

We test for the *sponge effect* by investigating the performance for the two first evaluations of each model, occurring after 10 and 35 minibatches of size 25 for vision, and after 5 and 11 training epochs for audio. We average the performance over the first two evaluations, as well as over all available datasets. We use MobileNetV2 for vision and the average of a 4-layer CNN and VGGish for audio. (The inception and design of our AutoDL for audio methods are presented in [Tuggener et al. \[2020\]](#)). The results are presented in Figure [6.3](#), they clearly show that the *sponge effect* is present in the vision and audio models.

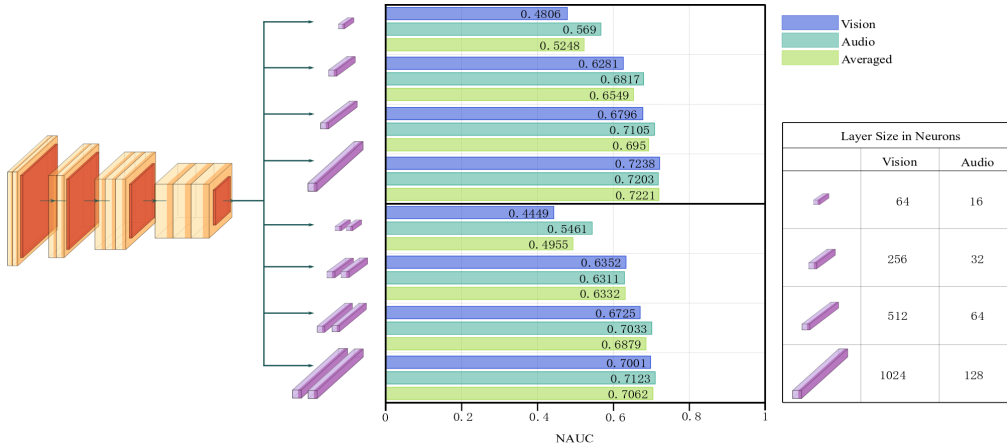


Figure 6.3. Sponge effect in vision and audio: Evidence for the increased sample efficiency of larger fully connected classifiers (“sponge effect”), shown by the averaged performance of eight different classifier designs on Vision and Audio data.

Even though the *sponge effect* is very apparent in the first training stage, the classifier size correlates much less with the final performance of a model. This observation demonstrates that increasing the classifier predictability (size) is not a free win, but there is a trade-off between training speed and proneness of a model to overfitting. Hence, the combination of large classifiers with sample-efficient regularization techniques, such as fast AutoAugment [Lim et al., 2019] or early stopping, is a promising step forward.

6.5.4 Intermediate conclusions

Our evaluations yield observations that may help optimise a vision system for AutoDL in an anytime-performance-oriented setting. Making models smaller makes them faster and therefore suited for fast learning. We could show that for certain hyperparameters (e.g., learning rate, pooling), a relatively simple evaluation can conclusively reveal the best choice for a resource-constrained setting. We also established that having a large classifier increases training speed through better sample efficiency. This rather counter-intuitive *sponge effect* can lead to bigger models training faster than smaller ones and is further discussed in Section 6.5.3.

Chapter 7

Robust neural network design: Is it enough to optimize CNN architectures on ImageNet?

Classification performance based on ImageNet is the de facto standard metric for CNN development. In this chapter, we challenge the notion that CNN architecture design solely based on ImageNet leads to generally effective convolutional neural network (CNN) architectures that perform well on a diverse set of datasets and application domains. To this end, we investigate and ultimately improve ImageNet as a basis for deriving such architectures. We conduct an extensive empirical study for which we train 500 CNN architectures, sampled from the broad AnyNetX design space, on ImageNet as well as 8 additional well-known image classification benchmark datasets from a diverse array of application domains. We observe that the performances of the architectures are highly *dataset dependent*. Some datasets even exhibit a negative error correlation with ImageNet across all architectures. We show how to significantly increase these correlations by *utilizing ImageNet subsets restricted to fewer classes*. These contributions can have a profound impact on the way we design future CNN architectures and help alleviate the tilt we see currently in our community with respect to over-reliance on one dataset.

This chapter is based on [Tugener et al. \[2022\]](#), previously published in Frontiers in Computer Science as a journal paper.

7.1 Introduction

Deep convolutional neural networks (CNNs) are the core building block for most modern visual recognition systems and have led to major breakthroughs in many domains of computer perception in the past several years. Therefore, the community has been searching the high dimensional space of possible network architectures for models with desirable properties. Important milestones such as DanNet [Ciresan, Meier, Masci, Gambardella and Schmidhuber, 2011a], AlexNet [Krizhevsky et al., 2012], VGG [Simonyan and Zisserman, 2015], HighwayNet [Srivastava et al., 2015a], and ResNet [He et al., 2016] (a HighwayNet with open gates) can be seen as update steps in this stochastic optimization problem and stand testament that the manual architecture search works. It is of great importance that the right metrics are used during the search for new neural network architectures. Only when we measure performance with a truly meaningful metric is it certain that a new high-scoring architecture is also fundamentally better. So far, the metric of choice in the community has generally been the performance on the most well-known benchmarking dataset—ImageNet [Russakovsky et al., 2015].

More specifically, it would be desirable to construct such a metric from a solid theoretical understanding of deep CNNs. Due to the absence of a solid theoretical basis novel neural network designs are tested in an empirical fashion. Traditionally, model performance has been judged using accuracy point estimates [Krizhevsky et al., 2012; Zeiler and Fergus, 2014; Simonyan and Zisserman, 2015]. This simple measure ignores important aspects such as model complexity and speed. Newer work addresses this issue by reporting a curve of the accuracy at different complexity settings of the model, highlighting how well a design deals with the accuracy versus complexity tradeoff [Xie et al., 2017; Zoph et al., 2018].

Very recent work strives to improve the quality of the empiric evaluation even further. There have been attempts to use extensive empirical studies to discover general rules on neural network design [Hestness et al., 2017; Rosenfeld et al., 2020; Kaplan et al., 2020b; Tuggenier et al., 2020], instead of simply showing the merits of a single neural network architecture. Another line of research aims to improve empiricism by sampling whole populations of models and comparing error distributions instead of individual scalar errors [Radosavovic et al., 2019].

We acknowledge the importance of the above-mentioned improvements in the empirical methods used to test neural networks, but identify a weak spot that runs through the above-mentioned work: the heavy reliance on ImageNet [Russakovsky et al., 2015] (and to some extent the very similar Cifar100 [Krizhevsky et al., 2009]). In 2011, Torralba and Efros already pointed out that visual recognition

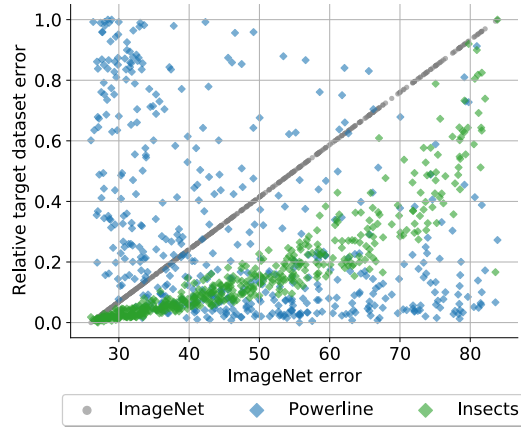


Figure 7.1. Is a CNN architecture that performs well on ImageNet automatically a good choice for a different vision dataset? This plot suggests otherwise: It displays the relative test errors of 500 randomly sampled CNN architectures on three datasets (ImageNet, Powerline, and Insects) plotted against the test error of the same architectures on ImageNet. The architectures have been trained from scratch on all three datasets. Architectures with low errors on ImageNet also perform well on Insects, on Powerline the opposite is the case.

datasets that were built to represent the visual world tend to become a small world in themselves [Torralba and Efros, 2011]. Objects are no longer in the dataset because they are important, they are important because they are in the dataset. In this chapter, we investigate how well ImageNet represents a diverse set of visual classification datasets—and present methods to improve said representation, such that CNN architectures optimized on ImageNet become more effective on visual classification beyond ImageNet. Specifically, we show: (a) an extensive empirical study examining the fitness of ImageNet as a basis for deriving generally effective CNN architectures; (b) we show how class-wise subsampled versions of ImageNet in conjunction with the original datasets yield a 2.5-fold improvement in average error correlations with other datasets (c) we identify cumulative block depth and width as the architecture parameters most sensitive to changing datasets.

As a tool for this investigation we introduce the notion of architecture and performance relationship (APR). The performance of a CNN architecture does not exist in a vacuum, it is only defined in relation to the dataset on which it is used. This dependency is what we call APR induced by a dataset. We study the change in APRs between datasets by sampling 500 neural network architectures and training all of them on a set of datasets. We then compare errors of the same architectures across datasets, revealing the changes in APR (see Figure 7.1).

This approach allows us to study the APRs induced by different datasets on a whole population of diverse network designs rather than just a family of similar architectures such as the ResNets [He et al., 2016] or MobileNets [Howard et al., 2017]. For code, sampled architectures, and complete training run data refer to the section B in the Appendix.

7.2 Related work

Neural network design. With the introduction of the first deep CNNs [Ciresan, Meier, Masci, Gambardella and Schmidhuber, 2011a; Krizhevsky et al., 2012] the design of neural networks immediately became an active research area. In the following years many improved architectures were introduced, such as VGG [Simonyan and Zisserman, 2015], Inception [Szegedy et al., 2015], HighwayNet [Srivastava et al., 2015a], ResNet [He et al., 2016] (a HighwayNet with open gates), ResNeXt [Xie et al., 2017], or MobileNet [Howard et al., 2017]. These architectures are the result of manual search aimed at finding new design principles that improve performance, for example increased network depth and skip connections. More recently, reinforcement learning [Zoph et al., 2018], evolutionary algorithms [Real et al., 2019] or gradient descent [Liu, Simonyan and Yang, 2019] have been successfully used to find suitable network architectures automatically. Our work relates to manual and automatic architecture design because it adds perspective on how stable results based on one or a few datasets are.

Empirical studies. In the absence of a solid theoretical understanding, large-scale empirical studies are the best tool at our disposal to gain insight into the nature of deep neural networks. These studies can aid network design [Greff et al., 2017; Collins et al., 2017; Novak et al., 2018] or be employed to show the merits of different approaches, for example that the classic LSTM [Hochreiter and Schmidhuber, 1997] architecture can outperform more modern models [Melis et al., 2018], when it is properly regularised. More recently, empirical studies have been used to infer more general rules on the behaviour of neural networks such as a power-law describing the relationship between generalization error and dataset size [Hestness et al., 2017] or scaling laws for neural language models [Kaplan et al., 2020b].

Generalization in neural networks. Despite their vast size have deep neural networks shown in practice that they can generalize extraordinarily well to unseen data stemming from the same distribution as the training data. To facilitate the best generalization properties researches often try to find the lowest-complexity

NNs that can still solve the problem adequately i.e. they address the bias vs. variance dilemma [Geman et al., 1992]. Such networks are often devised via weight penalty terms [Hanson and Pratt, 1988; MacKay, 1992] or pruning of existing high-complexity networks [Ash, 1989; Moody, 1991; Ivakhnenko, 1968; LeCun, Denker and Solla, 1989]. Why neural networks generalize so well is still an open and very active research area [Kawaguchi et al., 2017; Dinh et al., 2017; Zhang et al., 2017]. This work is not concerned with the generalization of a trained network to new data, but with the generalization of the architecture design progress itself. Does an architecture designed for a certain dataset, e.g. natural photo classification using ImageNet, work just as well for medical imaging? There has been work investigating the generalization to a newly collected test set, but in this case the test set was designed to be of the same distribution as the original training data [Recht et al., 2019].

Neural network transferability It is known that the best architecture for ImageNet is not necessarily the best base architecture for other applications such as semantic segmentation [Long et al., 2015] or object detection [Chen et al., 2019]. Researchers who computed a taxonomy of multiple vision tasks identified that the similarities between tasks did not depend on the used architecture [Zamir et al., 2019]. Research that investigates the relation between model performance on ImageNet and new classification datasets in the context of transfer learning [Razavian et al., 2014; Donahue et al., 2014] suggests that there is a strong correlation which is also heavily dependent on the training regime used [Kornblith et al., 2019]. Our work differs from the ones mentioned above in that we are not interested in the transfer of learned features but transfer of the architecture designs and therefore we train our networks from scratch on each dataset. Moreover do we not only test transferability on a few select architectures but on a whole network space.

Neural network design space analysis. Radosavovic et al. [Radosavovic et al., 2019] introduced network design spaces for visual recognition. They define a design space as a set of architectures defined in a parametric form with a fixed base structure and architectural hyperparameters that can be varied, similar to the search space definition in neural architecture search [Zoph et al., 2018; Real et al., 2019; Liu, Simonyan and Yang, 2019]. The error distribution of a given design space can be computed by randomly sampling model instances from it and computing their training error. We use a similar methodology but instead of comparing different design spaces, we compare the results of the same design space on different datasets.

7.3 Datasets

To enable cross dataset comparison of APRs we assembled a corpus of datasets. We chose datasets according to the following principles: (a) include datasets from a wide spectrum of application areas, such that generalization is tested on a diverse set of datasets; (b) only use datasets that are publicly available to anyone to ensure easy reproducibility of our work. Figure 7.2 shows examples and Table 7.1 lists meta-data of the chosen datasets. More detailed dataset specific information is given in the remainder of this section.

Concrete Özgenel and Sorguç [2018] contains 40 thousand image snippets produced from 458 high-resolution images that have been captured from various concrete buildings on a single campus. It contains two classes, positive (which contains cracks in the concrete) and negative (with images that show intact concrete). With 20 thousand images in both classes the dataset is perfectly balanced.

MLC2008 Shihavuddin et al. [2013] contains 43 thousand image snippets taken from the MLC dataset [Beijbom et al., 2012], which is a subset of the images collected at the Moorea Coral Reef Long Term Ecological Research site. It contains images from three reef habitats and has nine classes. The class distribution is very skewed with crustose coralline algae (CCA) being the most common by far (see Figure A.7 in Appendix A.1.1).

ImageNet Russakovsky et al. [2015] (ILSVRC 2012) is a large scale dataset containing 1.3 million photographs sourced from flickr and other search engines. It contains 1000 classes and is well balanced with almost all classes having exactly 1300 training and 50 validation samples.

HAM10000 Tschandl et al. [2018] is comprised of 10 thousand dermatoscopic images, collected from different populations and by varied modalities. It is a representative collection of all important categories of pigmented lesions that are categorized into seven classes. It is imbalanced with an extreme dominance of the melanocytic nevi (nv) class (see Figure A.7 in Appendix A.1.1).

Powerline Yetgin et al. [2017] contains images taken in different seasons as well as weather conditions from 21 different regions in Turkey. It has two classes, positive (that contain powerlines) and negative (which do not). The dataset contains 8000 images and is balanced with 4000 samples per classes.

Insects Hansen et al. [2019] contains 63 thousand images of 291 insect species. The images have been taken of the collection of British carabids from the Natural History Museum London. The dataset is not completely balanced but the majority of classes have 100 to 400 examples.

Intel Image Classification Bansal [2018] dataset (“natural”) is a natural

Table 7.1. Meta data of the used datasets.

Dataset	No. Images	No. Classes	Img. Size
Concrete	40K	2	227×227
MLC2008	43K	9	312×312
ImageNet	1.3M	1000	256×256
HAM10000	10K	7	296×296
Powerline	8K	2	128×128
Insects	63K	291	296×296
Natural	25k	6	150×150
Cifar10	60k	10	32×32
Cifar100	60k	100	32×32

scene classification dataset containing 25 thousand images and 6 classes. It is very well balanced with all classes having between 2.1 thousand and 2.5 thousand samples in the training set.

Cifar10 and Cifar100 [Krizhevsky et al., 2009] both consist of 60 thousand images. The images are sourced from the 80 million tiny images dataset [Torralba et al., 2008] and are therefore of similar nature (photographs of common objects) as the images found in ImageNet, besides the much smaller resolution. Cifar10 has 10 classes with 6000 images per class, Cifar100 consists of 600 images in 100 classes, making both datasets perfectly balanced.

7.4 Experiments and results

7.4.1 Experimental setup

We sample our architectures from the very general AnyNetX [Radosavovic et al., 2020] parametric network space. The networks in AnyNetX consist of a stem, a body, and a head. The body performs the majority of the computation, stem and head are kept fixed across all sampled models. The body consists of four stages, each stage i starts with a 1×1 convolution with stride s_i , the remainder is a sequence of d_i identical blocks. The blocks are standard residual bottleneck blocks with group convolution [Xie et al., 2017], with a total block width w_i , bottleneck ratio b_i and a group width g_i (into how many parallel convolutions the total width is grouped into). Within a stage, all the block parameters are shared. See Figure 7.3 for a comprehensive schematic. All models use batch

Table 7.2. Dataset-specific experimental settings.

Dataset	No. training epochs	Eval. error
concrete	20	top-1
MLC2008	20	top-1
Imagenet	10	top-5
HAM10000	30	top-1
Powerline	20	top-1
Insects	20	top-5
Natural	20	top-1
Cifar10	30	top-1
Cifar100	30	top-5

normalisation.

The AnyNetX design space has a total of 16 degrees of freedom, having 4 stages with 4 parameters each. We obtain our model instances by performing log-uniform sampling of $d_i \leq 16$, $w_i \leq 1024$ and divisible by 8, $b_i \in 1, 2, 4$, and $g_i \in 1, 2, \dots, 32$. The stride s_i is fixed with a stride of 1 for the first stage and a stride of 2 for the rest. We repeatedly draw samples until we have obtained a total of 500 architectures in our target complexity regime of 360 mega flops (MF) to 400 MF. We chose a narrow band of complexities to allow for fair comparisons of architectures with minimal performance variation due to model size. We use a very basic training regime, input augmentation consists of only flipping, cropping and mean plus variance normalisation, based on each datasets statistics. For training we use SGD with momentum and weight decay.

The same 500 models are trained on each dataset until the loss is reasonably saturated. The exact number of epochs has been determined in preliminary experiments and depends on the dataset (see Table 7.2). For extensive ablation studies ensuring the empirical stability of our experiments with respect to Cifar10 performance, training duration, training variability, top-1 to top-5 error comparisons, overfitting and class distribution see chapters A.1.1.1 to A.1.1.6 in Appendix A.1.1. Supplementary results on the effect of pretraining and the structure of the best performing architectures can be found in chapters A.1.2.1 and A.1.2.2 in Appendix A.1.2.

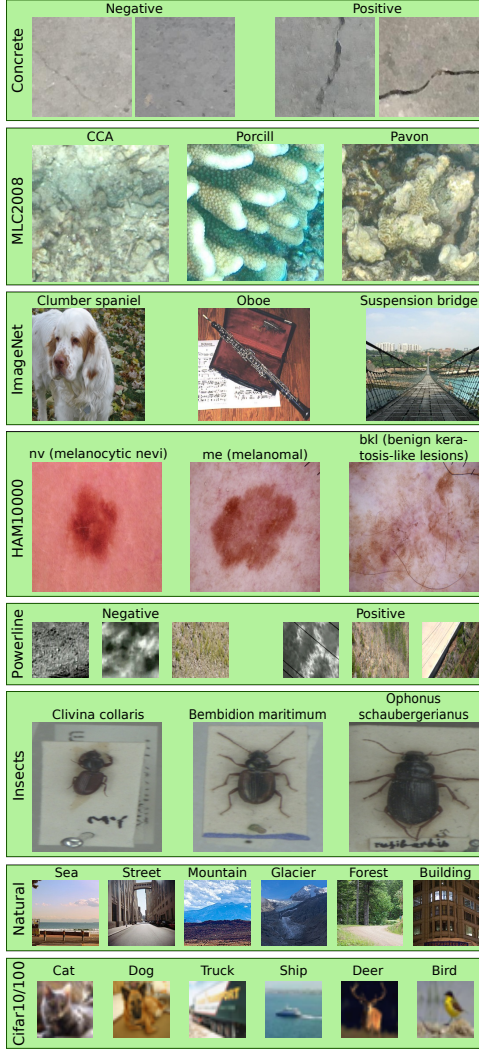


Figure 7.2

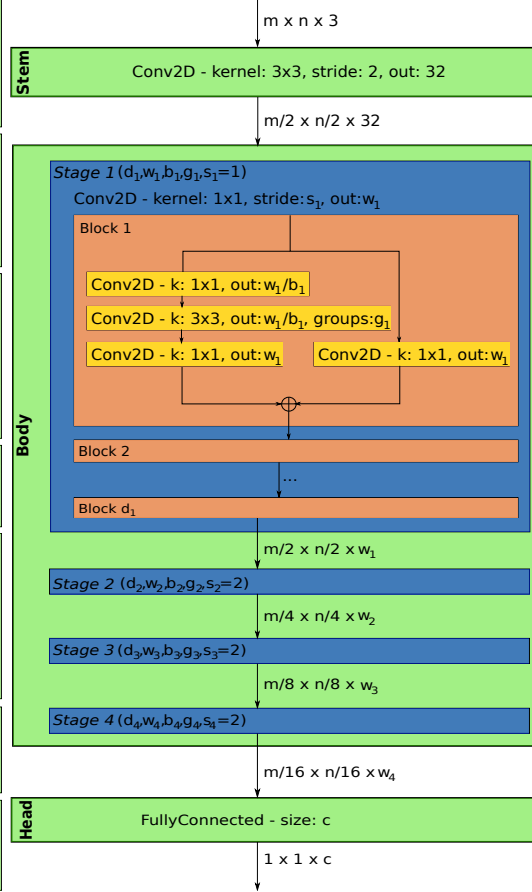


Figure 7.3

Figure 7.4. (A) Example images from each dataset. Images of Cifar10/100 are magnified fourfold, the rest are shown in their original resolution (best viewed by zooming into the digital document). (B) The structure of models in the AnyNetX design space, with a fixed stem and a head, consisting of one fully-connected layer of size c , (where c is the number of classes). Each stage i of the body is parametrised by d_i, w_i, b_i, g_i , the strides of the stages are fixed with $s_1 = 1$ and $s_i = 2$ for the remainder.

7.4.2 Experimental results

We analyze the architecture-performance relationship (APRs) in two ways. For every target dataset (datasets which are not ImageNet) we plot the test error of every sampled architecture against the test error of the same architecture (trained and tested) on ImageNet, visualizing the relationship of the target dataset’s APR with the APR on ImageNet. Second, we compute Spearman’s ρ rank correlation coefficient [Freedman et al., 2007]. It is a nonparametric measure for the strength of the relation between two variables (here the error on the target datasets with the error of the same architecture on ImageNet). Spearman’s ρ is defined on $[-1, 1]$, where 0 indicates no relationship and -1 or 1 indicates that the relationship between the two variables can be fully described using only a monotonic function.

Figure 7.8 contains the described scatterplots with the corresponding correlation coefficients in the title. The datasets plotted in the top two rows show a strong (Insects) or medium (MLC2008, HAM10000, Cifar100) error correlation with ImageNet. This confirms that many classification tasks have an APR similar to the one induced by ImageNet, which makes ImageNet performance a decent architecture selection indicator for these datasets. The accuracies on Concrete are almost saturated with errors between 0 and 0.5, it is plausible that the variations in performance are due to random effects rather than any properties of the architectures or the dataset, especially so since the errors are independent of their corresponding ImageNet counterparts. Therefore we refrain from drawing any further conclusions from the experiments on Concrete. This has implications for practical settings, where in such cases suitable architectures should be chosen according to computational and model complexity considerations rather than ImageNet performance, and reinforces the idea that practical problems may lie well outside of the ImageNet visual world [Stadelmann et al., 2018]. The most important insight from Figure 7.8, however, is that some datasets have a slight (Cifar10) or even strong (Powerline, Natural) *negative error correlation* with ImageNet. Architectures which perform well on ImageNet tend perform sub-par on these datasets. A visual inspection shows that some of the very best architectures on ImageNet perform extraordinarily poor on these three datasets. We can conclude that the APRs can vary wildly between datasets and high-performing architectures on ImageNet do not necessarily work well on other datasets.

An analysis of the correlations between all datasets (see Figure A.11 in Appendix A.1.2) reveals that Powerline and Natural not only have low correlation with ImageNet but also with most of the other datasets making these two truly particular datasets. Interestingly is the correlation between Powerline and Natural relatively high, which suggests that there is a common trait that makes these

two datasets behave differently. MLC 2008, HAM10000 and Cifar100 have a correlation of 0.69 with each other which indicates that they induce a very similar APR. This APR seems to be fairly universal since MLC 2008, HAM10000 and Cifar100 have a moderate to high correlation with all other datasets.

7.4.3 Impact of the number of classes

Having established that APR varies heavily between datasets, leaves us with the questions if it is possible to identify properties of the datasets themselves that influences its APR and if it is possible to control these factors to reduce the APR differences.

ImageNet has by far the largest number of classes among all the datasets. Insects, which is the dataset with the second highest class count, also shows the strongest similarity in APR to ImageNet. This suggests that the number of classes might be an important property of a dataset with respect to APR. We test this hypothesis by running an additional set of experiments on subsampled versions of ImageNet. We create new datasets by randomly choosing a varying number of classes from ImageNet and deleting the rest of the dataset (see Section 3. in the supplementary material for chosen classes). This allows us to isolate the impact of the number of classes while keeping all other aspects of the data itself identical. We create four subsampled ImageNet versions with 100, 10, 5, and 2 classes, which we call ImageNet-100, ImageNet-10, ImageNet-5, and ImageNet-2, respectively. We refer to the resulting group of datasets (including the original ImageNet) as the ImageNet-X family. The training regime for ImageNet-100 is kept identical to the one of ImageNet, for the other three datasets we switch to top-1 error and train for 40 epochs, to account for the smaller dataset size. (see section 7.4.3.1 in Appendix A.1.1 for a control experiment that disentangles the effects of reduced dataset size and reduced number of classes)

Figure 7.5 shows the errors on the subsampled versions plotted against the errors on original ImageNet. APR on ImageNet-100 shows an extremely strong correlation with APR on ImageNet. This correlation significantly weakens as the class count gets smaller. ImageNet-2 is on the opposite end has errors which are practically independent from the ones on ImageNet. This confirms our hypothesis that the number of classes is a dataset property with significant effect on the architecture to performance relationship.

We have observed that the number of classes has a profound effect on the APR associated with ImageNet-X members. It is unlikely that simply varying the number of classes in this dataset is able to replicate the diversity of APRs present in an array of different datasets. However, it is reasonable to assume that a

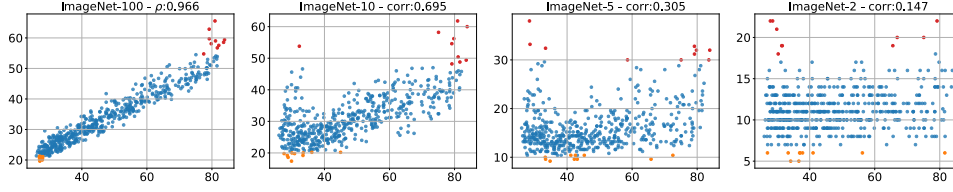


Figure 7.5. Error of all 500 sampled architectures on subsampled (by number of classes) versions of ImageNet (y-axis) plotted against the error of the same architectures on regular ImageNet (x-axis). The top 10 performances on the target dataset are plotted in orange and the worst 10 performances in red.

dataset’s APR is better represented by the ImageNet-X member closest in terms of class count, instead of ImageNet. We thus recreate Figure 7.8 with the twist of not plotting the target dataset errors against ImageNet, but against the ImageNet-X variant closest in class count (see Figure 7.6). We observe gain in correlation across all datasets, in the cases of MLC2008 or Cifar10 a quite extreme one. The datasets which have a strong negative correlation with ImageNet (Powerline, Natural) have slightly (Natural) or even moderately (Powerline) positive correlation to their ImageNet-X counterparts. A visual inspection shows that the best models on Imagenet-X also yield excellent results on Powerline and Natural, which was not the case for ImageNet. Table 7.3 shows the error correlations of all target datasets with ImageNet as well as with their ImageNet-X counterpart. The move from ImageNet to ImageNet-X more than doubles the average correlation (from 0.19 to 0.507), indicating that the ImageNet-X family of datasets is capable to represent a much wider variety of APRs than ImageNet alone.

7.4.3.1 Disentangling the effects of class count and dataset size

We showed how sub-sampled versions of ImageNet matching the number of classes of the target dataset tend to represent the APR of said target dataset far better. A side effect of downsampling ImageNet to a specific number of classes is that the total number of images present in the dataset also shrinks. This raises the question if the increase in error correlation is actually due to the reduced dataset size rather than to the matching class count. We disentangle these effects by introducing another downsampled version of ImageNet, Imagenet-1000-10. It retains all 1000 classes but only 10 examples per class resulting in a dataset with the same number of classes as ImageNet but with the total number of images of ImageNet-10. We train our population of architectures on ImageNet-1000-10 and show the error relationship of Cifar10, Natural, and Powerline with ImageNet-

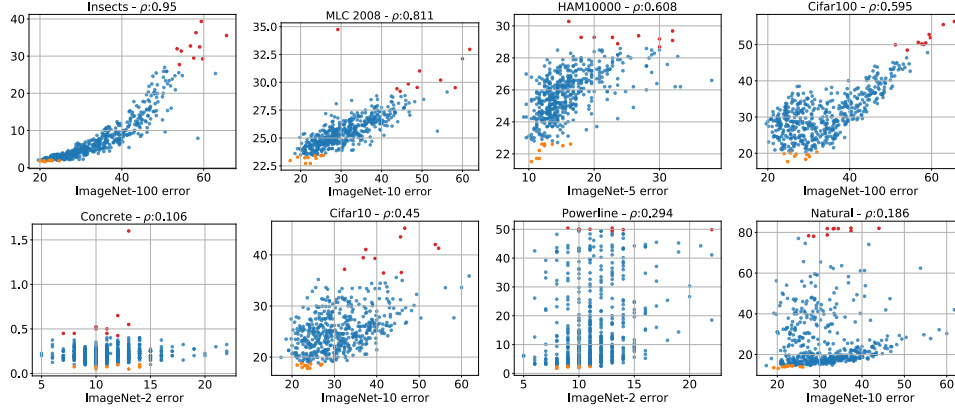


Figure 7.6. Test errors of all 500 sampled architectures on target datasets (y-axis) plotted against the test errors of the same architectures on the ImageNet-X (x-axis). The top 10 performances on the target dataset are orange, the worst 10 performances red.

1000-10 (as well as with ImageNet and ImageNet-10 as a reminder) in Figure 7.7. The plots show that there are some correlation gains by using ImageNet-1000-10 over ImageNet, but the effect is far lower compared to ImageNet-10. *This shows that downsampling size has a minor positive effect but the majority of the gain in APR similarity achieved through class downsampling actually stems from the reduced the class number.*

7.4.4 Identifying drivers of difference between datasets

The block width and depth parameters of the top 15 architectures for ImageNet (see Figure A.10 in Appendix A.1.2) follow a clear structure: they consistently start with low values for both block depth and width in the first stage, then the values steadily increase across the stages for both parameters. The error relationships observed in Figure 7.8 are consistent with how well these patterns are replicated by the other datasets. Insects shows a very similar pattern, MLC2008 and HAM10000 have the same trends but more noise. Powerline and Natural clearly break from this structure, having a flat or decreasing structure in the block width and showing a quite clear preference for a small block depth in the final stage. Cifar10 and Cifar100 are interesting cases, they have the same behaviour as ImageNet with respect to block width but a very different one when it comes to block depth.

We thus investigate the effect of the cumulative block depth (summation of the depth parameter for all four stages, yielding the total depth of the architecture)

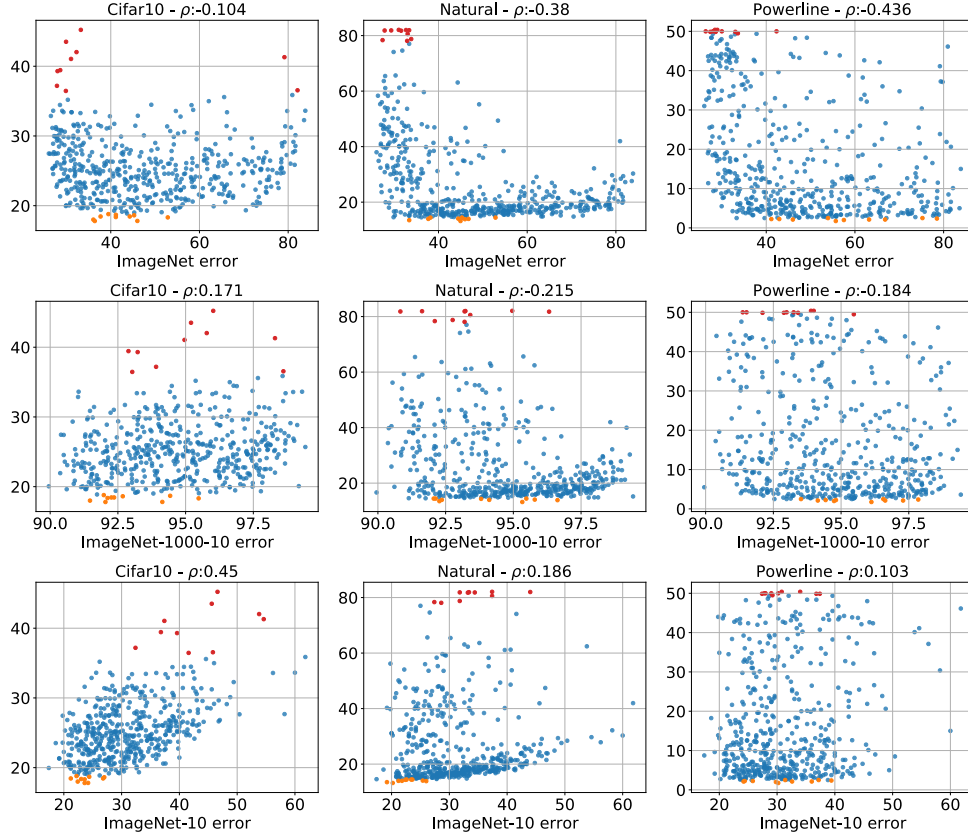


Figure 7.7. The errors of all 500 architectures on Cifar10, Natural, and Powerline plotted against the errors on ImageNet (top row), ImageNet-1000-10 (middle row) and ImageNet-10 (bottom row). We observe that class-wise downsampling has the largest positive effect on error correlation.

Table 7.3. Comparison of error correlations between target datasets and ImageNet as well as the closest ImageNet-X member.

Dataset	ρ -ImageNet	ρ -ImageNet-X	Difference
concrete	0.001	0.106	0.105
MLC2008	0.476	0.811	0.335
ham10000	0.517	0.608	0.091
powerline	-0.436	0.294	0.73
insects	0.967	0.95	-0.017
natural	-0.38	0.186	0.566
cifar10	-0.104	0.45	0.554
cifar100	0.476	0.595	0.119
Average	0.19	0.507	0.317

across the whole population of architectures by plotting the cumulative block depth against the test error for the six above-mentioned datasets. Additionally, we compute the corresponding correlation coefficients. Figure 7.9 shows that the best models for ImageNet have a cumulative depth of at least 10. Otherwise there is no apparent dependency between the ImageNet errors and cumulative block depth. The errors of Insects do not seem to be related to the cumulative block depth at all. HAM10000 has a slight right-leaning spread leading to a moderate correlation, but the visual inspection shows no strong pattern. The errors on Powerline, Natural, and Cifar100 on the other hand have a strong dependency with the cumulative block depth. The error increases with network depth for all three datasets. with the best models all having a cumulative depth smaller than 10.

We also plot the cumulative block widths against the errors and compute the corresponding correlation coefficients for the same six datasets (see Figure 7.10). We observe that the ImageNet errors are negatively correlated with the cumulative block width, and visual inspection shows that a cumulative block width of at least 250 is required to achieve a decent performance. The errors on Insects and HAM10000 replicate this pattern to a lesser extent, analogous to the top 15 architectures. Powerline and Natural have no significant error dependency with the cumulative block width, but Cifar100 has an extremely strong negative error dependency with the cumulative block width, showing that it is possible for a dataset to replicate the behaviour on ImageNet in one parameter but not the other. In the case of Cifar100 and ImageNet, low similarity in block depth

and high similarity in block width yield a medium overall similarity of ARPs on Cifar100 and Imagenet. This is consistent with the overall relationship of the two datasets displayed in Figure 7.8

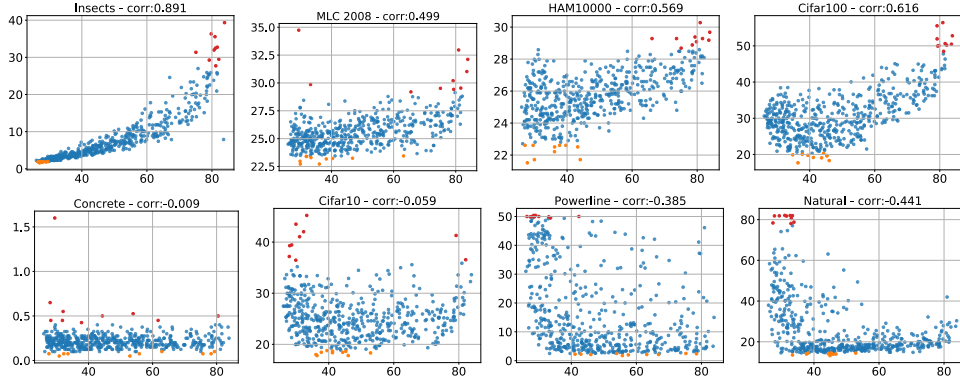


Figure 7.8. Test errors of all 500 sampled architectures on target datasets (y-axis) plotted against the test errors of the same architectures (trained and tested) on ImageNet (x-axis). The top 10 performances on the target datasets are plotted in orange and the worst 10 performances in red.

Combining this result with the outcome of the last section, we study the interaction between the number of classes, the cumulated block depth and the cumulative block width. Table 7.4 contains the correlations between cumulative block depth/width and the errors on all members of ImageNet-X. With decreasing number of classes, the correlation coefficients increase for cumulative block depth and cumulative block width. Although the effect on cumulative block depth is stronger, there is a significant impact on both parameters. We therefore can conclude that both optimal cumulative block depth and cumulative block width can drastically change based on the dataset choice and that both are simultaneously influenced by the class count.

7.5 Discussion and intermediate conclusions

ImageNet is not a perfect proxy. We have set out to explore how well other visual classification datasets are represented by ImageNet. Unsurprisingly there are differences between the APRs induced by the datasets. More surprising and worrying, however, is that for some datasets ImageNet not only is an imperfect proxy but a very bad one. The negative error correlations with Natural, Powerline and Cifar10 indicates that architecture search based on ImageNet performance is worse than random search for these datasets.

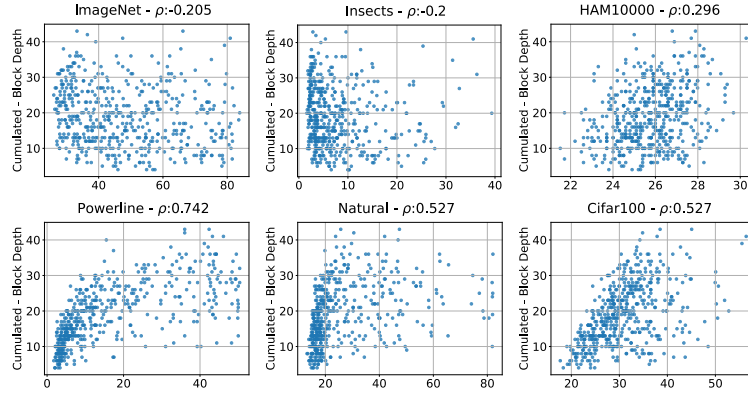


Figure 7.9. depths

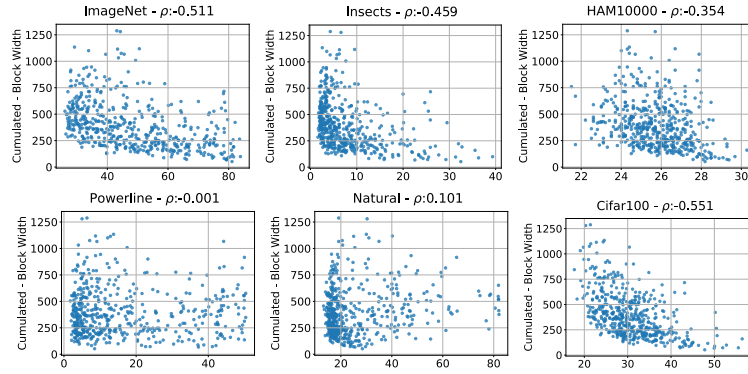


Figure 7.10. widths

Figure 7.11. Errors of all 500 sampled architectures on ImageNet, Insects, HAM10000, Powerline, Natural, and Cifar100 (x-axis) plotted against the cumulative block (A) depths and (B) depths (y-axis).

Table 7.4. Correlation of observed error rates with the cumulative block depth and width parameters for all ImageNet-X datasets.

Dataset	C. Block Depth	C. Block Width
ImageNet	−0.205	−0.511
ImageNet-100	−0.022	−0.558
ImageNet-10	0.249	−0.457
ImageNet-5	0.51	−0.338
ImageNet-2	0.425	−0.179

Varying the number of classes is a cheap and effective remedy. It is striking how much more accurately the ImageNet-X family is able to represent the diversity in APRs present in our dataset collection, compared to just ImageNet by itself. It has become commonplace to test new architectures in multiple complexity regimes [He et al., 2016; Howard et al., 2017]. We argue for augmenting this testing regime with an additional dimension for class count. This simple and easy to implement extension would greatly extend the informative value of future studies on neural network architectures.

Visual variability is less important than anticipated. In the introduction we critiqued the over-reliance on ImageNet based on the limits of "visual world" it represents, since it only contains natural images and is mostly focused on animals and common objects. However, our results show that datasets with visually very different content such as Insects and HAM10000 have a high APR correlation with ImageNet. For Natural and Cifar10, which contain natural images, the opposite is the case. This shows that the visual domain of a dataset is not the central deciding factor for choosing the correct CNN architecture.

Future directions. A future similar study should shed light on how well the breadth of other domains such as object detection, segmentation or speech classification are represented by their essential datasets. If the representation is also insufficient it could be verified if the symptoms are similar and the varying the number of classes also helps covering more dataset variability in these domains. A labeled dataset will always be a biased description of the visual world, due to having a fixed number of classes and being built with some systematic image collection process. Self-supervised learning of visual representations [Jing and Tian, 2019] could serve as a remedy for this issue. Self-supervised architectures could be fed with a stream of completely unrelated images collected from an arbitrary number of sources in a randomized way. A comparison of visual features learned in this way could yield a more meaningful measure of the quality of CNN

architectures.

Limitations As with any experimental analysis of a highly complex process such as training a CNN it is virtually impossible to consider every scenario. We list below three dimensions along which our experiments are limited together with measures we took to minimize the impact of these limitations.

Data scope: We criticize ImageNet for only representing a fraction of the “visual world”. We are aware that our dataset collection does not span the entire “visual world” either but went to great lengths to maximise the scope of our dataset collection by purposefully choosing datasets from different domains, which are visually distinct.

Architecture scope: We sample our architectures from the large AnyNetX network space. It contains the CNN building blocks to span basic designs such as AlexNet or VGG as well as the whole ResNet, ResNeXt and RegNet families. We acknowledge that there are popular CNN components not covered, however, Radosavovic et al. [Radosavovic et al., 2020] present ablation studies showing that network designs sourced from high performing regions in the AnyNetX space also perform highly when swapping in different originally missing components such as depthwise convolutions [Chollet, 2017], swish activation functions [Ramachandran et al., 2018] or the squeeze-and-excitation [Hu et al., 2018] operations.

Training scope: When considering data augmentation and optimizer settings there are almost endless possibilities to tune the training process. We opted for a very basic setup with no bells and whistles in general. For certain such aspects of the training, which we assumed might skew the results of our study (such as training duration, dataset preprocessing etc.), we have conducted extensive ablation studies to ensure that this is not the case (see sec. A.1.1.2 and A.1.1.6 in Appendix).

Chapter 8

Efficient rotation invariance in computer vision tasks: Artificial mental rotation

In this Chapter we present *artificial mental rotation* (AMR), a novel deep learning paradigm for dealing with in-plane rotations inspired by the neuro-psychological concept of mental rotation. Our simple AMR implementation works with all common CNN and ViT architectures. We test it on ImageNet, Stanford Cars, and Oxford Pet. With a top-1 error (averaged across datasets and architectures) of 0.743, AMR outperforms the current state of the art (rotational data augmentation, average top-1 error of 0.626) by 19%. We also easily transfer a trained AMR module to a downstream task to improve the performance of a pre-trained semantic segmentation model on rotated CoCo from 32.7 to 55.2 IoU.

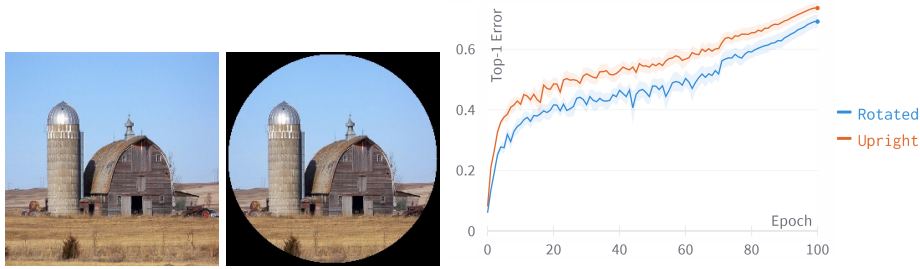


Figure 8.1. An example image from ImageNet (left) and a version of the same image with its corners masked, to allow for non-obvious and reversible rotations (middle). The top-1 classification error on ImageNet aggregated across 8 CNN architectures for upright (orange) and rotated (blue) training, revealing a training slow-down for rotated inputs (right).

This chapter is based on Tugener et al., [2024], under review at AAAI 2025.

8.1 Introduction

Natural vision systems in humans and animals are able to recognize objects irrespective of transformations such as rotations and translations as well as the observer's point of view, all of which can have a tremendous impact on the appearance of said object. This is a highly desirable property for all vision systems, especially if they are to be deployed in a real-world setting [Stadelmann et al., 2018, 2019]. Both CNNs [Fukushima et al., 1983] and transformers [Vaswani et al., 2017] (a.k.a. neural fast weight programmers [Schmidhuber, 1992; Schlag et al., 2021]) inherently integrate translational invariance into their design. For rotations, this is not the case and both methods perform very poorly when facing inputs at an unusual angle. This can be exploited for adversarial attacks [Engstrom et al., 2019] and causes serious issues in applications where rotated inputs are common. There are currently two main research avenues trying to alleviate this issue:

One is focused on building architectures that incorporate rotational invariance or sometimes equivariance directly into the neural network design. [Cohen and Welling, 2016] introduced Group Equivariant Convolutional Neural Networks (G-CNNs) which are equivariant to a discrete symmetry group of rotations. [Marcos et al., 2017] on the other hand proposed to rotate the convolutional filters instead of the representations. A common drawback among rotation equivariant neural networks is that the memory footprint grows linearly with the angular resolution, severely limiting their practical use.

Another much more widely used approach is based on input data augmentation. The data is rotated at training time such that the model can learn all appearances of an object. This yields good results and can scale to any problem size. It is, however, still an inefficient method because the different appearances of a single object are learned individually, which artificially inflates the complexity of any given problem. This results in slower training and, consequently, lower final performance (see Figure 8.1).

It is a long-standing conjecture in neuro-psychology that when humans try to identify an object, they mentally simulate rotations of that object with the goal of matching it to an internal representation (i.e. they perform mental rotation). [Shepard and Metzler, 1971] were the first to formally study this phenomenon. They were able to show that the time human subjects need to determine whether pairs of 3D figures have the same shape grows linearly with the angle of rotation between the two objects. This strongly suggests that humans perform mental rotation; otherwise, the re-identification task would be completed in constant time across angles.

Inspired by this finding, we propose a third way for deep neural networks to deal with rotated inputs that we dub *artificial mental rotation* (AMR). The core idea of AMR is to first find the angle of rotation of a given input and then rotate it back to its canonical appearance before further processing, thus performing an artificial version of mental rotation. This has the advantage that the underlying method of visual recognition itself does not have to be hardened against rotations, therefore all models (even trained ones) can be used in conjunction with an AMR module without any altering.

In short, the core contributions of this chapter are: (a) We introduce the concept of mental rotation to deep learning, (b) we present a simple neural network architecture that implements AMR and can be paired with all common CNNs and ViTs, (c) we extensively test the merits of AMR on ImageNet, Stanford Cars, and Oxford Pet, and conclude that it significantly outperforms data augmentation, the current state-of-the-art, (d) we present AMR results on MNIST to enable the comparison with computationally expensive alternatives, (e) we study the viability of AMR in a scenario where only parts of the test data are rotated, (f) we present comprehensive ablation studies proofing that our trained AMR modules work in practice on synthesized as well as naturally rotated data, and (g) we show the easy transferability of a trained AMR module by moving it to a downstream task (in this case semantic segmentation), significantly increasing the performance of an existing model on rotated data.

8.2 Related work

There are ongoing efforts to incorporate rotation invariance (or in some cases equivariance) directly into the architectures of deep neural networks, especially for CNNs. [Dieleman et al. \[2015\]](#) introduced a rotation invariant CNN system for galaxy morphology prediction that uses multiple rotated and cropped snippets of the same image as input. [Cohen and Welling \[2016\]](#) presented G-CNNs which are equivariant to a larger number of symmetries such as reflections or rotations. This is achieved by lifting the network features to a desired symmetry group. [Romero and Cordonnier \[2021\]](#) presented group equivariant vision transformers by extending the symmetry group lifting concept to self-attention. [Worrall et al. \[2017\]](#) introduce H-Nets which replace regular CNN filters using circular harmonics. Alternatively, [Marcos et al. \[2017\]](#) have proposed to rotate the filters of a CNN and then apply spatial and orientation pooling to reduce and merge the resulting features. [Laptev et al. \[2016\]](#) introduce TI-pooling, which allows pooling of the CNN outputs for an arbitrary number of different angled versions of

the same input to create an equivariant feature. All these methods share the key drawback that their memory footprint grows linearly with the angular resolution, which severely limits their practical usability.

Data augmentation [Baird, 1992] is very widely used to improve the robustness and generalizability of vision models [Simard et al., 2003]. It can even be used to harden the model against adversarial attacks [Shafahi et al., 2019]. Data augmentation has also been shown to be very effective for rotated inputs [Quiroga et al., 2020]. Data augmentation is the current de-facto standard technique for dealing with rotated data since it is easy to use and effective. However, data augmentation is not efficient because different appearances of the same object are learned independently.

There have been previous attempts to leverage the concept of mental rotation for computer vision. Ding and Taylor [2014] trained a factored gated restricted Boltzmann machine to actively transform pairs of examples to be maximally similar in a feature space. Boominathan et al. [2016] train a shallow neural network to classify if an image is upright. They then combine this with a Bayesian optimizer to find upright images. They use this setup to improve image retrieval robustness. In the space of 3D vision a mental rotation-based approach achieved state-of-the-art performance for rotated point cloud classification [Fang et al., 2020].

8.3 Artificial mental rotation module

Our AMR approach requires three components. First, a base model (BM) is required, for which any common CNN or ViT [Dosovitskiy et al., 2021] architecture can be used. There is no need to modify the BM in any way, hence the BM can generally be sourced in a fully (pre-)trained form. Additionally, it requires a rotation algorithm designed for images; here we use the method available in OpenCV [Bradski, 2000]. The last necessary component is the AMR module itself, presented in this section. Due to their differing designs, CNNs and ViTs use slightly varying AMR modules, described in Sections 8.3.1 and 8.3.2.

AMR training While training the AMR module, the BM is frozen such that its classification performance is not disturbed. For the training, we use datasets, like ImageNet, where the objects are typically shown in an upright position. Under this constraint, we can employ self-supervised training by randomly rotating the input images and asking the AMR module to recover the angle we previously applied.

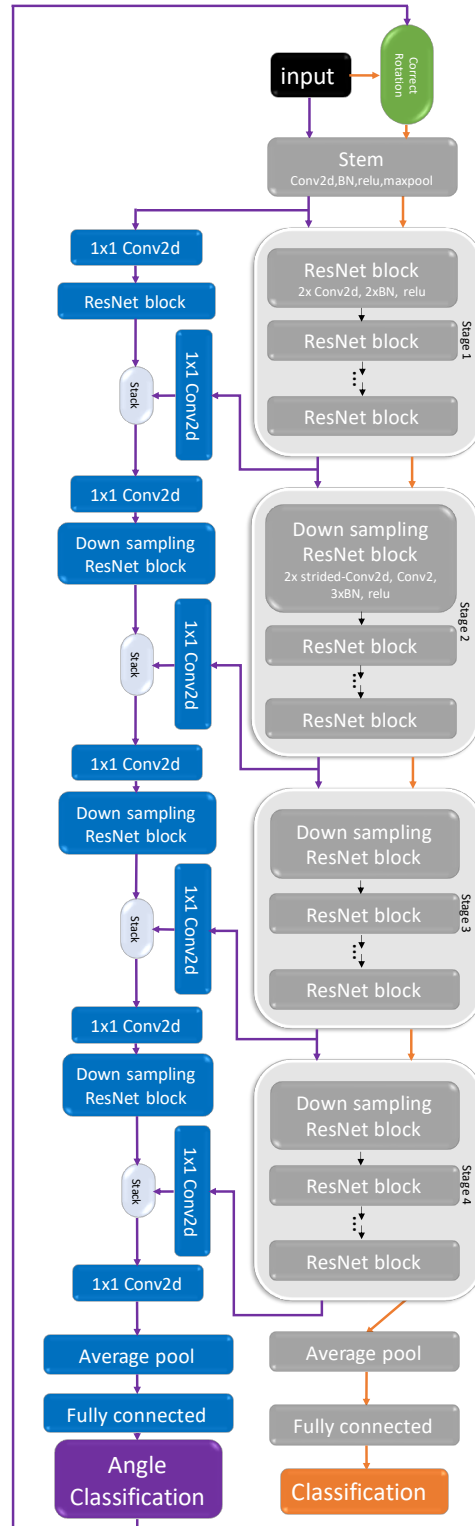


Figure 8.2. Architecture of our artificial mental rotation module for CNNs. The base CNN, in this case a ResNet, is shown in grey. The components of the AMR module are shown in blue. The information flow in stage 1 (angle classification) is purple while the information flow in stage 3 (image classification) is shown in orange.

AMR inference AMR inference is performed in a three-step process: (1) The input’s angle is classified by running it through the BM and the AMR module. (2) The input is rotated by the negative amount of the angle determined in step one. (3) The rotation-corrected input is processed by the BM.

Step (3) is identical to AMR-free inference since the BM is frozen during AMR training and there is no information flow through the AMR module during this step. Therefore AMR could also be framed as a preprocessing method by reducing it to steps (1) and (2).

8.3.1 AMR module for CNNs

Our AMR module is designed as an add-on to a given BM (see Figure 8.2), so it can repurpose the features computed by the BM and only requires a small number of additional weights. Features are copied into the AMR module at five different BM stages. In the case of ResNe(X)ts [He et al., 2016; Xie et al., 2017] (a.k.a. Highway Nets with open gates [Srivastava et al., 2015a]) this happens directly after the stem and after each of the four ResNe(X)t stages. For EfficientNets [Tan and Le, 2019] we use the end of stages 2, 4, 6, and 8 as extraction points. When the copied features enter the AMR module they are first processed by a single 1×1 2D convolution to compress the feature depth. For all but the first AMR module stages, these features are then stacked with the output of the previous AMR module stage followed by another 1×1 2D convolution to half the feature depth of the stack. Only then the data is processed by a single ResNet block. After the last stage, we employ average pooling and a single fully connected layer with 360 outputs to create the angle prediction.

8.3.2 AMR module for ViTs

The AMR module for ViTs is very similar to the one for CNNs. We again extract features at five different locations. For ViT-16-b these are after encoder blocks 1, 4, 7, and 12. Since there is no spatial downsampling in ViTs there is no advantage in processing the extracted features in stages. We, therefore, stack them all at once followed by a single 1×1 2D convolution. This stack is then processed by four ViT encoder modules. Lastly, we extract the same classification token that was used in the BM and apply a fully connected layer for the angle classification.

Table 8.1. ImageNet top-1 accuracies. Upright testing (up) of the upright trained base model is assumed to be the performance ceiling (% ceil). Average (by angle) rotated accuracies (rot) are given for the upright and rotated trained base models as for AMR 33 epochs and AMR 5 epochs.

Testing	Upright Training			Rotated Training		AMR 33		AMR 5	
	up	rot	% ceil	rot	% ceil	rot	% ceil	rot	% ceil
ResNet-18	0.695	0.433	0.62	0.598	0.86	0.676	0.97	0.666	0.96
ResNet-50	0.768	0.537	0.70	0.673	0.88	0.755	0.98	0.746	0.97
ResNet-152	0.779	0.552	0.71	0.730	0.94	0.767	0.98	0.760	0.98
EfficientNet-b0	0.680	0.454	0.67	0.611	0.90	0.666	0.98	0.656	0.96
EfficientNet-b2	0.692	0.467	0.67	0.612	0.88	0.678	0.98	0.669	0.97
EfficientNet-b4	0.710	0.485	0.68	0.618	0.87	0.696	0.98	0.689	0.97
ResNext-50-32x4d	0.773	0.551	0.71	0.686	0.89	0.761	0.98	0.754	0.98
ResNext-101-32x8d	0.785	0.571	0.73	0.728	0.93	0.772	0.98	0.766	0.98
ViT-16b	0.691	0.459	0.66	0.503	0.73	0.669	0.97	0.664	0.96
Average	0.730	0.501	0.69	0.640	0.87	0.716	0.98	0.708	0.97

8.3.3 Motivation for add-on design

We opted to design our AMR module as an add-on to existing base networks because we conjecture that the features that have been trained for classification will also be at least partly useful for angle detection and the AMR module can profit from the training resources that have already been invested into the base network. This design choice therefore allows for an AMR module that consists of very few layers on its own and thus can be trained very quickly. We confirm this conjecture with an ablation study (see Appendix [A.2.2](#)).

8.4 Experiments

We aim to showcase the merits of AMR on natural images. Therefore, we test it on ImageNet (ILSVRC 2012) [[Russakovsky et al., 2015](#)] and verify our results on Stanford Cars [[Krause et al., 2013](#)] and Oxford Pet [[Parkhi et al., 2012](#)]. To ensure that artificial rotations are not obvious, we mask out the corners of all images such that a centred circle remains (see Figure [8.1](#) and Section [8.5](#) for an ablation study ensuring the artificial rotations are not carrying any unwanted information). For a fair comparison between upright training (without data augmentation) and training with random rotations as input data augmentation (rotated training),

we train all of our base models from scratch with this masking applied. For all of our training runs, we use image normalization based on dataset statistics. No further data augmentation is applied to keep the experiments as simple as possible (except, of course, input rotation for the rotated training models). To obtain representative results we replicate our experiments on a variety of base models. We use three different ResNets, three EfficientNets, and two ResNeXts for a total of eight CNN architectures. On ImageNet we also employ a vision transformer in the form of ViT-16b, which is unsuited for the other smaller datasets. For each upright trained base model, we train two AMR modules: One is trained for one-third of the base model’s training time (in epochs) and the other one for one-twentieth.

Training details For all base models, we use the implementations from the torchvision [maintainers and contributors, 2016] Python package without any modifications. To enable optimal training speed our code is based on the ffcv library [Leclerc et al., 2022]. All training details and links to code and trained model weights can be found in Appendix B.3.

Testing We first evaluate the upright base models on upright data. We use these performances as the ceiling of what can be achieved on rotated data. Then we test the upright and rotated base models as well as the AMR-enhanced models for rotated performance by rotating the test set two degrees at a time and running a full evaluation for each angle. We present the resulting data visually in polar plots (see Figure 8.3 and Figure 8.4) as well as in table form (see Table 8.1 and Table 8.2) by averaging across angles.

Table 8.2. Stanford Cars and Oxford Pet top-1 accuracies averaged across all architectures, columns are analogous to the Table 8.1 shown above for ImageNet.

Testing	Upright Training			Rotated Training		AMR 300		AMR 50	
	up	rot	% ceil	rot	% ceil	rot	% ceil	rot	% ceil
Stanford Cars	0.867	0.165	0.19	0.618	0.71	0.796	0.92	0.746	0.86
Oxford Pet	0.741	0.483	0.65	0.603	0.81	0.712	0.96	0.670	0.90

ImageNet We train all of our base CNNs for 100 epochs on ImageNet, and the vision transformer is trained for 300 epochs, in accordance with the training recipes for the torchvision base models. We then train two AMR modules in

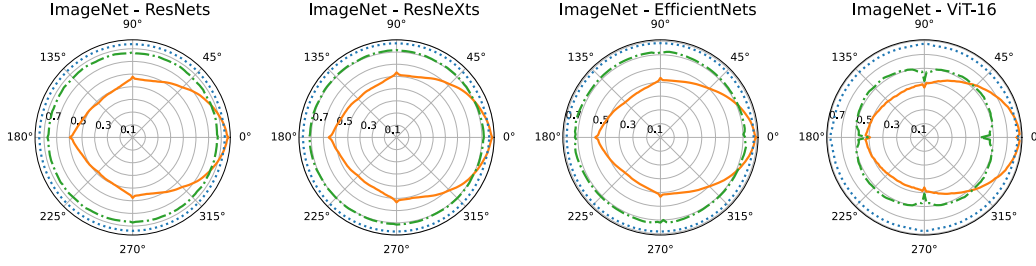


Figure 8.3. Polar plots of ImageNet top-1 accuracies by angle, averaged across architectures (except ViT). The performances of the upright base models are shown in solid orange, the rotated training base models are shown in dash-dotted green, and AMR performance (averaged across both epoch regimes) is shown in dotted blue lines.

conjunction with each upright trained base model, one for 33 epochs and the other for 5. Table 8.1 contains the top-1 accuracies on rotated data for all models. Additionally, the ceiling accuracy is also reported (upright data with upright trained model). As suspected, there is a steep drop in accuracy between upright and rotated testing for the upright-trained models, both for the CNNs as well as the ViT. On average only 69 percent of the ceiling performance (% ceil) is retained. The models which have been trained with random rotations fare much better, they achieve 87% ceil. It is noteworthy that the ViT only rises from 66 to 73 of the ceiling performance. This makes sense since ViTs tend to be less sample efficient compared to CNNs and therefore suffer more from the increased problem complexity caused by the random rotations. AMR-33 achieves 98% ceil, significantly outperforming rotated training. AMR-5 is slightly worse with 97% ceil, but it shows that it is possible to obtain an AMR module that is very useful with minimal training resources. Figure 8.3 contains polar plots that show the ImageNet top-1 accuracies of the different architecture families by angle. The solid orange lines show the accuracies of the upright-trained base models. We observe that the accuracies have their highest points at zero degrees rotation and then symmetrically drop off with increasing angle, reaching their lowest points at 135 and 225 degrees. We further observe that rotated training (green dash-dotted line) and AMR (blue dotted line) both achieve rotational invariance and exhibit performances that are independent of test time angles. Corresponding to the reported results in Table 8.1, AMR performance is consistently better than rotated training.

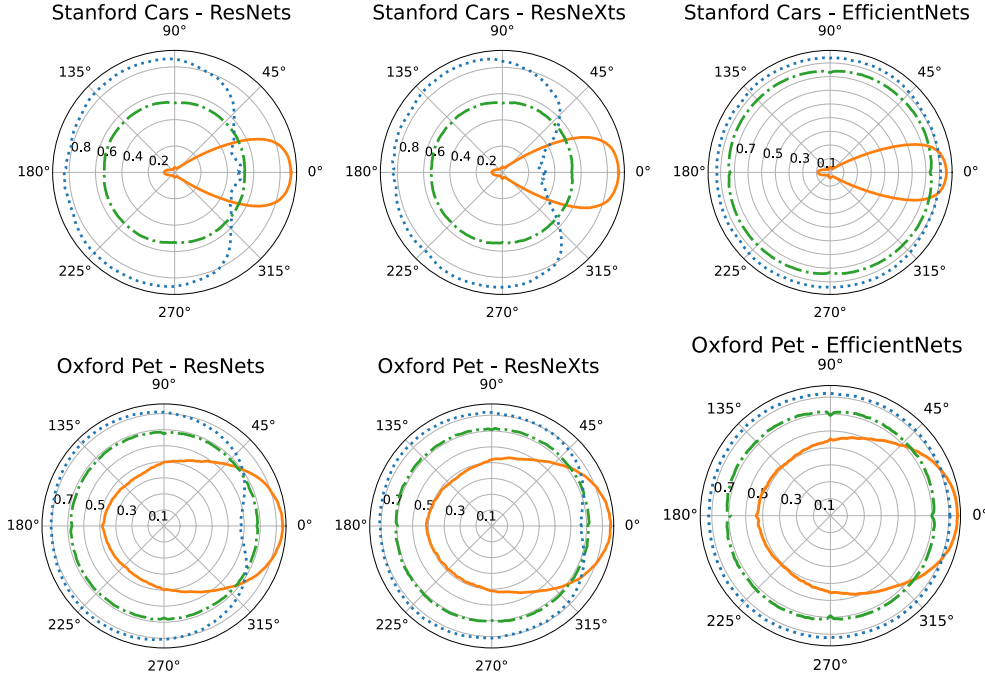


Figure 8.4. Polar plots of Stanford Cars (top row) and Oxford Pet (bottom row) top-1 accuracies analogous to the ones shown above for ImageNet (see Figure 8.3).

Stanford Cars and Oxford Pet Due to Stanford Cars and Oxford Pet being smaller datasets we forgo ViTs and train the CNN models for more epochs on Stanford Cars and Oxford Pet. On Stanford Cars we train the base models for 1000 epochs, and the corresponding AMR modules are trained for 300 and 50 epochs, respectively. On Oxford Pet, we train the base models for 3000 epochs and the AMRs for 1000 and 150 epochs. Table 8.2 shows the top-1 accuracies averaged across architectures and averaged across angles where appropriate (for full table see Table A.2 in the Appendix) and Figure 8.4 polar plots, analogous to the ones for ImageNet in the above paragraph. Our core findings are replicated on both datasets: Rotating the images reduces all the models’ performances and AMR remains the most potent way of addressing rotations. On Stanford Cars, the performance loss caused by rotations on the upright trained model is much more severe (19% ceil) compared to ImageNet (69% ceil), with the models failing almost completely when facing rotations larger than 20 degrees (see left column of Figure 8.4). This makes sense intuitively since cars are almost always upright in pictures with minimal variation, thus the models experience almost no variation during training. This is further supported by the observation that Oxford Pet,

Table 8.3. Top-1 accuracies of ResNets on upright (up) and rotated (rot) ImageNet, accompanied with breakpoints (BP) that signify the share of rotated data in the test set necessary for alternative methods (rotated training, AMR) to outperform upright training.

Testing	Upright Training		Rotated Training			AMR-33		
	up	rot	up	rot	BP	up	rot	BP
ResNet-18	69.5	43.3	59.3	58.9	35.5%	67.1	67.6	9.0%
ResNet-50	76.5	53.7	69.2	67.3	35.0%	75.1	75.5	6.1%
ResNet-152	77.9	55.2	73.6	73.0	19.5%	76.2	76.7	7.4%
Average	74.6	50.7	67.4	66.4	30%	72.8	73.3	7.5%

Table 8.4. Top-1 accuracies on rotated MNIST for ResNet-18 based methods as well as related works, accompanied by ResNet-18 upright top-1 accuracy as a baseline.

Method	Top-1 Acc.
ResNet-18 (upright - ceil performance)	0.996
ResNet-18	0.48
ResNet-18 + rotated training	0.978
ResNet18 + AMR	0.981
Harmonic Networks [Worrall et al., 2017]	0.983
Ti-pooling [Laptev et al., 2016]	0.988
G-CNNs [Cohen and Welling, 2016]	0.977
RotEqNet [Marcos et al., 2017]	0.989

which is also a small dataset but contains animals that are naturally less static compared to cars, exhibits a milder drop off (65% ceil). We further observe that on Stanford Cars and to a lower extent on Oxford Pet, EfficientNets perform much better than ResNe(X)ts on rotated data, both with rotated training and AMR, while all architectures perform roughly equally well on upright data. We conjecture this is because EfficientNets have been designed to be sample efficient. This could allow them to train filters useful for a wide variety of tasks (such as AMR) even on a small dataset and a relatively short training time. However, an unexpected result is that AMR paired with ResNe(x)t models showed a decline in performance when approaching 0 degrees, while EfficientNets do not suffer from this effect. See Appendix A.2.1 for further investigation of this phenomenon.

Comparison with existing rotation equivariant methods The focus of AMR is large datasets like ImageNet and beyond. The current literature for rotation equivariant methods is focused on computationally expensive methods that re-engineer the basic structure of the used neural networks (as mentioned in Section 8.2). Consequently, MNIST is the benchmark dataset of choice for most of these methods. We put our work into perspective with these related works by presenting the performances of ResNet-18, ResNet-18 + rotated training and ResNet18+AMR on MNIST (see Table 8.4). The ceiling performance of ResNet18 on upright MNIST is almost one, which is to be expected. Similar to the larger datasets above is the performance of AMR far superior to rotated training. Most importantly, the performance of ResNet18+AMR is comparable to the ones of the related works which are much narrower in scope. This shows that AMR not only exhibits the best performance on large datasets but also achieves state-of-the-art performance on a small dataset amongst highly specialized methods.

AMR usefulness given the prevalence of rotated data In an applied scenario, it is not always realistic that all inputs are presented at a random angle. We therefore investigate the usefulness of AMR when the test data consists of a combination of upright (up) and rotated (rot) images. To this end, we compute top-1 test errors on ImageNet of the ResNet family models on rotated and upright inputs separately. We repeat this process for upright training, rotated training and AMR-33 (see Table 8.3). We then linearly combine up and rot performances to obtain the final performances for mixed datasets consisting of both upright and rotated data. We increase the percentage of rotated data in the test mix until alternative methods (rotated training, AMR-33) start outperforming the default of upright training. We call percentages of parity between methods breakpoints (BP). Unsurprisingly, the BPs for rotated training (30% on average) are much higher than the ones of AMR-33 (7.5%). The key finding here is that BPs for AMR-33 are all below 10% which shows that only a small portion of the test set needs to be non-upright for AMR to be a worthwhile choice.

8.5 Validity of self-supervised training built on artificial rotation

Self-supervised learning based on artificial data modifications always warrants great caution. It is often unclear if the model learns to solve the desired task or if it simply learns to find unintended shortcuts in the self-supervision procedure. In

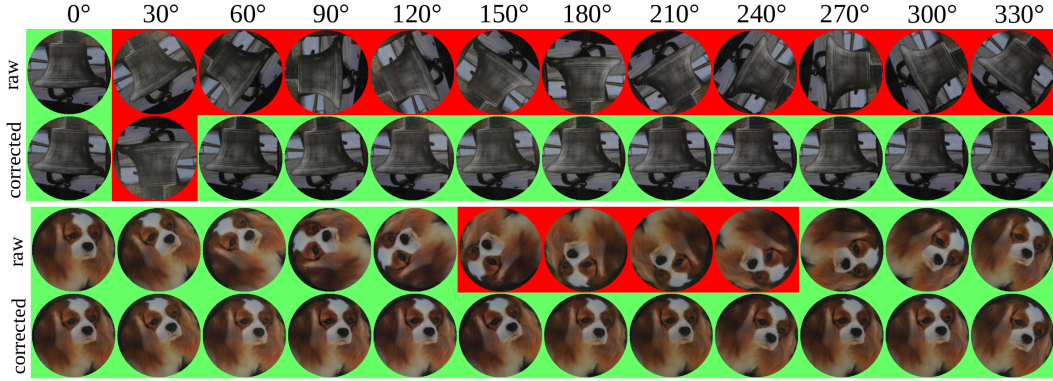


Figure 8.5. Photographs of two printed ImageNet validation samples taken at 12 different angles. Both samples are shown in their original state (raw) and after the mental rotation step (corrected). The color of the masked-out region indicates if the corresponding image has been correctly classified. The mental rotation and classification steps have been performed by ResNet50 + AMR33.

our case, we use a digital rotation algorithm on our input images. While none are visible to the human eye, algorithm-specific artefacts are introduced to the rotated images. This raises the question if the AMR module learns to classify the correct rotation angle based on unwanted traces of the rotation algorithm rather than by understanding the contents of the image. To ensure this is not the case, we perform the following ablation study: We print out seven images sourced from different classes from the ImageNet validation set. We then take photos of each of those printouts at twelve different in-plane rotations by physically rotating the print in 30-degree intervals. This way we naturally introduce rotation and can guarantee the absence of any rotation algorithm artifacts that the model could have learned to use. The images were printed using a Konica Minolta bizhub 450i on maximum resolution with guidelines to enable accurate angular distances (see Figure A.14 in the Appendix). The photos were then taken by hand using a Nikon Coolpix P7000 digital camera. Figure 8.5 shows all twelve re-digitized photos for two cases (raw). The color-coded background indicates if that photo was correctly classified by a standard trained ResNet50 base model (green denotes correct, red an error). We observe a similar effect as with Stanford Cars: Like a car, bells have a very clearly defined upright position. The bell, therefore, is only classified correctly when it is upright. Dogs on the other hand are very variable in appearance (e.g. head turned, laying down etc), thus the dog is only misclassified when it is completely upside down at rotations between 150 and 240 degrees. The second rows (corrected) show the outcome of applying Resnet50 + AMR33

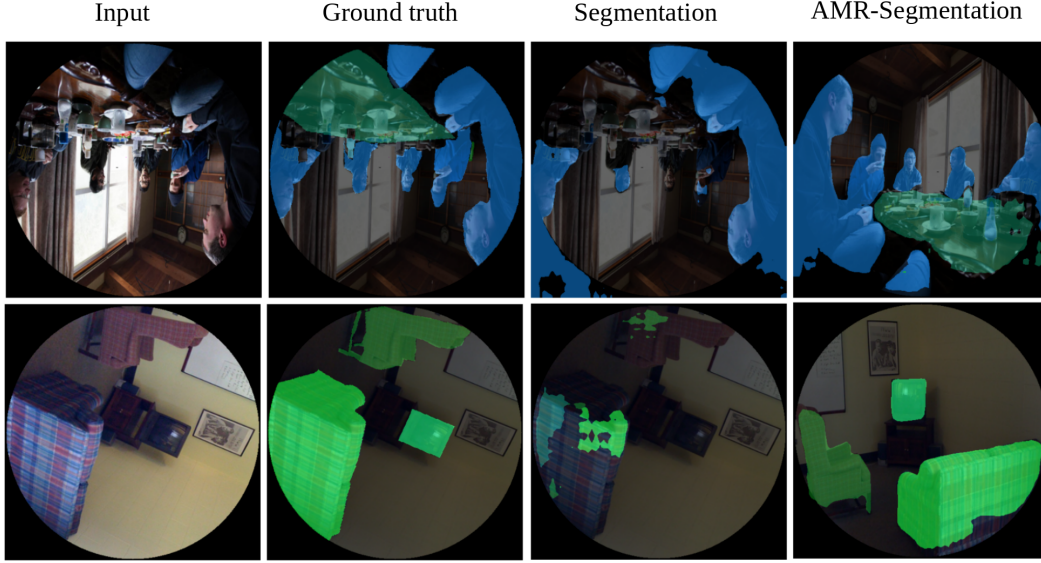


Figure 8.6. Two rotated examples (rows) from the CoCo validation set with their corresponding ground truth, segmentation output, and AMR-corrected segmentation output (columns). The segmentation on the rotated image exhibits bad performance in both samples. In row one, the persons are segmented fairly well but the table is missing fully, in row two the segmentation fails almost completely. The column AMR-segmentation shows the output for the images that have been un-rotated using AMR. The AMR module identified the correct rotation angle in both cases, which lead to significantly improved segmentation performance (again in both cases).

to the above photos. The AMR module is able to correct the orientation of all but one photo. We can therefore conclude that it learned to classify the angles by understanding the image contents rather than relying on artifacts introduced by the self-supervision process. We further observe that the rotation correction is much more precise in the bell case than for the dog. This ties in with our assumption that the network’s filters are much more precisely tuned to a sharp upright position for the bell compared to the dog. Across all 84 photos, the standard ResNet50 achieves a top-1 classification accuracy of 0.57. ResNet50 + AMR33, on the other hand, achieves a top-1 accuracy of 0.96, showing that the AMR module properly works on all printed images.

8.6 Application to a novel downstream task: semantic segmentation

Since they use the same neural network building blocks, the assumption that models for other vision tasks like object detection or semantic segmentation also struggle with rotated inputs suggests itself. In this section, we test this hypothesis and demonstrate how a trained AMR module can be used to easily improve the rotational stability of models for other tasks than classification. Here we choose semantic segmentation as an example. As the base model, we use a fully convolutional ResNet50 and source the matching pre-trained weights named 'FCN_ResNet50_Weights.COCO_WITH_VOC_LABELS_V1' from torchvision. These weights have been trained on MSCoCo-Stuff [Lin et al., 2014], with a reduced class set only containing the classes that are also available in PascalVOC [Everingham et al., 2010]. We again mask the corners of all images. On this data, the pre-trained model achieves a mean intersection over union (IoU) of 57.6. We then randomly rotate the images which causes the mean IoU to drop to 32.7. This confirms that not only object classification models but also semantic segmentation and likely most other vision models perform far worse when confronted with rotated inputs. We now take our ResNet50 + AMR33 which has been trained on ImageNet and use it to perform AMR steps (1) and (2) on CoCo without any additional retraining or modification. The angle-corrected inputs are then fed back into the base semantic segmentation model. This approach yields an IoU of 55.2, showing that AMR also works for semantic segmentation and that a trained AMR module can be easily transferred between similar datasets. Figure 8.6 shows two examples from the CoCo validation set with their corresponding ground truth, segmentation output, and AMR-corrected segmentation output visually confirming our findings. For both examples, the AMR module correctly inverted the angle of the image, which (again in both cases) significantly improves the segmentation performance.

8.7 Limitations and future work

A key drawback of AMR is that two forward passes are necessary. This is part of the core design and cannot be changed. It is mitigated partially by the fact that a smaller model can be chosen in conjunction with AMR and still outperform a large model trained with rotational data augmentation due to the inefficiency of that approach resulting in a less costly forward pass even at test time. With applicability in mind, we opted to focus on 2D in-plane rotations of whole images

featuring one dominant object. Our work is therefore not suited for cases where multiple objects are individually rotated. This scenario could be addressed by combining a region-proposal based method such as Faster-RCNN [Girshick, 2015] with AMR at proposal level. In the real world, 3D objects are rotated on two axes, which can lead to much more drastic changes in appearance. Extending AMR to 3D objects would be a very promising, most natural extension of this work. An exciting future application for AMR models would be reducing the rotational variability of an existing dataset (e.g. ImageNet, by making all appearing objects upright). This would further disentangle the training of upright appearances from rotations which would likely lead to improved training efficiency of base models.

8.8 Intermediate conclusions

We have presented AMR, a neuropsychology-inspired approach for handling rotated data in vision systems, novel to deep learning. We have shown that AMR consistently outperforms the current standard of input data augmentation across different deep architectures and datasets. We have shown the viability of AMR in realistic cases where the data is a mixture of upright and rotated inputs. We further presented a sanity check which confirms that our self-supervised learning setup learns to identify rotations by the content of the images and not by any artifacts introduced by the training procedure. Lastly, we have shown how a trained AMR module can easily be transferred to another model built for a different task (in our case semantic segmentation) to greatly improve its rotational stability.

Chapter 9

Conclusions

9.1 Summary

In the opening of this thesis, we have highlighted the challenges of applying cutting-edge deep learning methods to practical applications and promising research areas for addressing these challenges. In Chapter [3](#), we devised and built DeepScores, a high-quality dataset for optical music recognition comprised of 300'000 sheets of written music accompanied with ground truth for classification, segmentation, and detection containing roughly 80 million annotated objects in 118 classes. We built on DeepScores in Chapter [4](#) and introduced the Deep Watershed Detector, a novel object detection method, which is designed to handle the particular challenges of optical music recognition, such as the high density of many small objects on a high-resolution image. It significantly advanced the state of the art in optical music recognition by being the first method that was able to process full pages in an end-to-end fashion. In Chapter [5](#), we extended our findings from the previous chapter, ensuring good results in messy "real-world" scenarios. To this end, we improved our dataset. We released DeepScoresV2, which includes higher-level rhythm and pitch annotations, an increased character, and, most importantly, oriented bounding boxes, which fit slanted bars and ties much tighter. DeepScoresV2 was accompanied by baseline detectors, illustrating the advantages of oriented detection methods. Finally, we introduced ScoreAug, a simple but very effective data augmentation scheme based on stochastic noise and real-world perturbations sourced from "blank" scanned pages, greatly increasing the performance of trained models on low-quality data. In Chapter [6](#), we first surveyed the state of the art in automated machine learning. Then, we introduced Portfolio Hyberband, a novel automated machine learning method designed for time-sensitive and low-cost scenarios. We then investigated the anytime per-

formance of various CNN designs and extracted the Sponge Effect as a design paradigm for automated deep learning. Chapter 7 presented an explorative study on the relationship between the performance of model architectures and the used datasets. It shows that optimal architecture choice is heavily dataset-dependent, and it is therefore not wise to benchmark novel developments against ImageNet only. We further investigated drivers of difference between the datasets. We showed that using multiple subsets of ImageNet restricted to fewer classes already lead to a much more representative benchmarking of neural network architectures. In Chapter 8, we presented artificial mental rotation (AMR), a bio-inspired method for handling rotated inputs in deep neural networks. Our self-supervised reference implementation worked with CNN and transformer-based methods and exhibited significantly stronger performance than existing methods. We further demonstrated how a trained AMR module can be easily transferred to another related task and improve rotational stability out of the box.

9.2 Future directions

The work presented in this thesis contains proven strategies for successfully leveraging deep learning techniques in various practical scenarios. It also revealed many promising avenues for future work — so plentiful that we don't recount all of them here. The full details of future directions can be found in the intermediate conclusion sections in each chapter. Here, we highlight the four directions we deem especially fruitful, with the potential to fundamentally reshape the way certain problems are tackled today.

Higher order music information for optical music recognition

Our work on OMR focuses on detecting and classifying a predefined set of atomic music notation primitives. Although challenging the direct learning of higher level musical concepts (such as the allowed length of a bar, rhythm information of each note, etc.) would have big advantages. Forcing the model to gain a deeper understanding of musical concepts could, in turn, lead to improved recognition performance. It would simplify the challenging task of reassembling the musical primitives into a semantically sound piece of music. A promising way forward would be to attach higher-level information to the individual symbols, such as rhythm or connections between symbols. Continuously incorporating more musical knowledge into the detection method could ultimately lead to an image-to-sequence model that directly outputs a high-level music encoding such as music-XML.

Generative modelling for data augmentation

Our introduction of ScoreAug has shown that realistic distortions (as opposed to stochastic noise or filters) are key to learning document recognition models that generalize to documents of low quality. ScoreAug is constrained to a low number of real-world noise masks sourced from scans. This concept could be greatly extended by using generative modelling to learn how to generate useful and varied additive distortion masks.

A taxonomy for training datasets

We have shown that optimal model choice heavily depends on the data used. A meaningful taxonomy of datasets would allow hierarchical classification of datasets into groups of similar and distinct data. This enables the correlation of a new dataset with existing ones and thus expected model performance (i.e. gives a meaningful indication of which architecture works the best on a given dataset). This would mark a massive step forward, enabling researchers to achieve two feats: a) Building a landscape of public datasets and constructing a superset of diverse datasets representative of the whole breadth of relevant applications. This superset could be used to test and benchmark methods in a more dataset-agnostic way. b) Significantly improve automated machine learning. Classifying the dataset according to the taxonomy allows an informed selection of model architecture and hyperparameter.

Extending artificial mental rotation to three dimensions

We have introduced artificial mental rotation for the in-plane rotation of natural images. In the real world, we observe 3D objects along two axes. Rotation across two axes is necessary to cover all possible points of view on any given object. Additionally, it leads to much more drastic changes in appearances. Extending artificial mental rotation to this 3D case (e.g. by utilizing a video game engine) would mark a fundamental step forward for the concept. It would enable models to learn complete representations of an object in a much more natural way, optimally combined with active learning interacting with the positioning of the shown objects.

Appendix A

Extended Result

A.1 Is it enough to optimize CNN architectures on Imagenet?

A.1.1 Verifying the numerical robustness of our study

In this Chapter, we present additional studies designed to test for possible flaws or vulnerabilities in our experiments. We conduct these to further strengthen the empirical robustness of our results.

A.1.1.1 Stability of Empirical Results on Cifar10

The top-1 errors of our sampled architectures on Cifar10 lie roughly between 18 and 40, which is fairly poor, not only compared to the state of the art but also compared to performance that can be achieved with fairly simple models. This calls into question if our Cifar10 results are flawed in a way that might have lead us to wrong conclusions. We address this by running additional tests on Cifar10 and evaluate their impact on our main results. We get a goalpost for what performance would be considered good with our style of neural network and training setup by running the baseline code for Cifar10 published by Radosavovic et al. [Radosavovic et al., 2020]. Table A.1 shows that these baseline configurations achieve much lower error rates. We aim to improve the error results on Cifar10 in two ways: First we train our architecture population with standard settings for 200 epochs instead of 30, second we replaced the standard network stem with one that is specifically built for Cifar10, featuring less stride and no pooling. Figure A.1 shows scatterplots of the errors from all 500 architectures on Cifar10 against the errors on ImageNet and ImageNet-10. We can see that both

Table A.1. Top-1 error of reference network implementations [Radosavovic et al., 2020] for Cifar10.

Model	ResNet-56	ResNet-110	AnyNet-56	AnyNet-110
Error	5.91	5.23	5.68	5.59

new training methods manage to significantly improve the performance with a minimum top-1 error below 10 in both cases. More importantly can we observe that both new training methods have, despite lower overall error, a very similar error relationship to ImageNet. The error correlation is even slightly lower than with our original training (replicated in Figure A.1 left row). We can also see that in all three cases the error relationship can be significantly strengthened by replacing ImageNet with ImageNet-10, this shows that tuning for individual performance on a dataset does not significantly impact the error relationships between datasets which further strengthens our core claim.

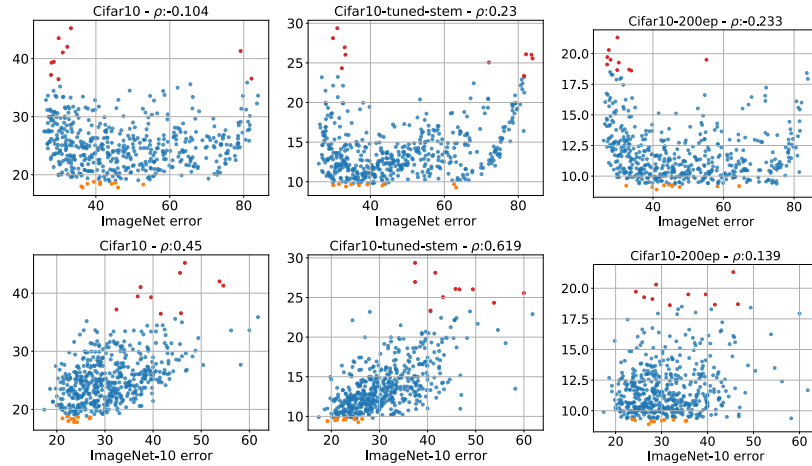


Figure A.1. The Cifar10 test errors of all 500 architectures plotted against ImageNet (top row) and ImageNet-10 (bottom row), shown for our original Cifar10 training (left column), training with a Cifar10 specific stem in the architecture (middle column), and training for 200 epochs, which is roughly 6 times longer (right column). The plots show that the error correlation with ImageNet-10 is much larger in all three cases, confirming that optimizing for individual Cifar10 performance does not alter our core result.

A.1.1.2 Verifying Training Duration

Since we have a limited amount of computational resources and needed to train a vast number of networks we opted to train the networks up to the number of epochs where they started to saturate significantly in our pre-studies. As we have seen in section [A.1.1.1](#) can the network performance still improve quite a bit if it is trained for much longer. Even though the improved performances on Cifar10 did not yield any results contradicting the findings of our study, we still deemed it necessary to closer inspect what happened in the later stages of training and thus performed a sanity check for Cifar10 as well as the other two datasets that show a negative error correlation with ImageNet—Powerline and Natural. Figure [A.2](#) shows the Cifar10 test error curves of 20 randomly selected architectures over 200 epochs. On the left side we see the same curves zoomed in to epochs 30 to 200. We see that the error decreases steadily for all architectures, the ranking among architectures barely changes past epoch 30. The relative performance between architectures and not absolute error rates are relevant for our evaluations, we can therefore conclude that the errors at epoch 30 are an accurate enough description of an architecture’s power.

For Powerline and Natural, we select the five best and five worst architectures respectively and continue training them for a total of five times the regular duration. Figure [A.3](#) shows the resulting error curves. Both datasets exhibit minimal changes in the errors of the top models. On Natural we observe clear improvements on the bottom five models but similar to Cifar10 there are very little changes in terms of relative performance. Powerline exhibits one clear cross-over but for the remainder of the bottom five models the ranking also stays intact. Overall we can conclude that longer training does not have a significant effect on the APR of our datasets.

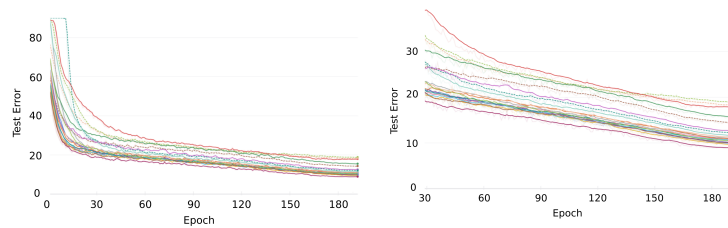


Figure A.2. Cifar10 test error curves of 20 randomly sampled architectures trained over 200 epochs (left). The same error curves but cut to epochs 30 to 200.

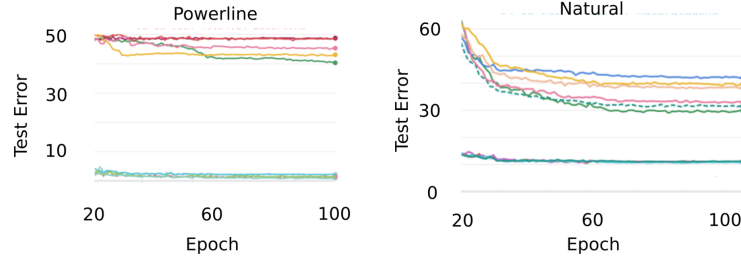


Figure A.3. Test error curves of the five best and five worst models on Powerline and Natural, respectively, when training is continued to epoch 100

A.1.1.3 Impact of Training Variability

The random initialization of the model weights has an effect on the performance of a CNN. In an empirical study it would therefore be preferable to train each model multiple times to minimize this variability. We opted to increase the size of our population as high as our computational resources allow, this way we get a large number of measurements to control random effects as well as an error estimate of a large set of architectures. However, we still wanted to determine how much of the total variability is caused by training noise and how much is due to changing the architectures. We estimate this by selecting two of the sampled CNN designs, number 147 performing slightly above average with an error of $e_{147} = 11.9$ and number 122 performing slightly below average with $e_{122} = 14.5$. The quantiles of the error distribution from all 500 architectures are $q_{0.25} = 11.53$, $q_{0.5} = 13.02$ and $q_{0.75} = 15.46$ with an overall mean of $\mu = 13.9$. We then train the architectures 147 and 122 each 250 times. Figure A.4 shows the error distributions of both selected architectures as well as the overall distribution obtained from training each of the 500 architectures once. There is of course some variability within both architectures but both individual architectures produce very narrow densities and show essentially no overlap. We can therefore conclude that the effect of choosing an architecture is much greater than the variability caused by random training effects.

A.1.1.4 Relationship of Top-1 with Top-5 Error on ImageNet, Insects and Cifar100

We opted to use top-5 error since it is the most widely reported metric for ImageNet and the top-5 numbers are therefore easy to interpret on that dataset. Many of our datasets have a significantly lower number of classes such that top-5 error makes little sense and we opted to use top-1 for those. This raises the question

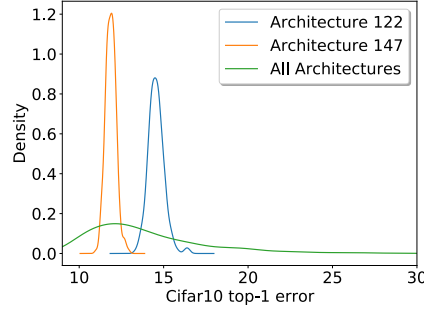


Figure A.4. Error distributions on Cifar10 of two architectures (122, 147) both trained from scratch 250 times as well as the Cifar10 error distribution of all 500 architectures. The plot shows that the variability caused by changing architecture is much larger than the one caused by random training effects.

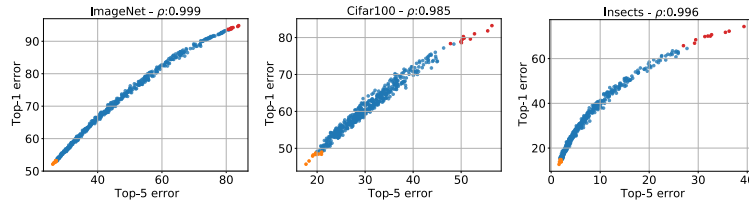


Figure A.5. Top-1 error plotted against top-5 error of all 500 architectures on ImageNet, Cifar100, and Insects. The plots reveal that on all three datasets the errors have a very close relationship: it is not perfectly linear but is monotonically ascending

if comparing top-1 with top-5 errors introduces unwanted perturbations into our analysis. We therefore compare the top-1 and top-5 errors for the three datasets on which we use top-1 error (see Figure [A.5](#)). We see that the two metrics have an almost linear relationship for the ImageNet and Cifar100 datasets. More importantly are the top-1 to top-5 error graphs monotonically ascending for all three datasets, such that the ordering of architectures does not change when swapping between the two metrics. Since we are interested in the relative performances of our sampled architectures changing between top-1 and top-5 error does not impact our analysis.

A.1.1.5 Overfitting of High-Capacity Architectures

The best architectures on Powerline, Natural and Cifar100 have a very small cumulated depth, so it is only natural to ask if the deeper architectures perform poorly due to overfitting. We address this concern by plotting the *training errors* of

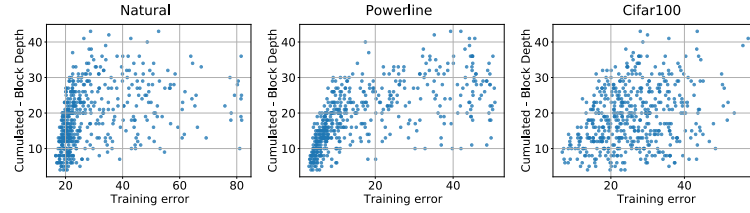


Figure A.6. Training errors of the sampled architectures (x-axis) plotted against the cumulated block depth for the 3 datasets that have the lowest test errors on shallow architectures. We observe that for all three datasets shallow architectures also have the lowest training errors. Therefore overfitting is not the cause of this behaviour.

Powerline, Natural, and Cifar100 against the cumulative block depths (see Figure A.6). The training errors are strongly correlated with the cumulative block depth, just like the test errors. Plots of the cumulated block depth show almost the same structure for training and test errors. We can therefore exclude overfitting as a reason why the shallower networks perform better on Powerline, Natural, and Cifar100.

A.1.1.6 Impact of Class Distribution

MLC2008 and HAM1000 have a strong class imbalance. They both have one class which makes up a large amount of the dataset. In order to study the impact of an imbalanced class distribution, we created two new more balanced datasets out of the existing data the following way: we reduced the number of samples in the overrepresented class such that it has the same amount of samples as the second most common class. We call these datasets MLC2008-balanced and HAM10000-balanced. Their new class distributions can be seen in Figure A.7. We train our architecture population on MLC2008-balanced and HAM10000-balanced leaving the training configuration otherwise unaltered. Figure A.8 shows the errors on the balanced datasets versus the errors on the unbalanced counterparts.

For both HAM10000 and MLC2008, there is a strong correlation between the errors on the balanced and unbalanced datasets. We can therefore conclude that class imbalance is not a determining factor for the APRs of HAM10000 or MLC2008.

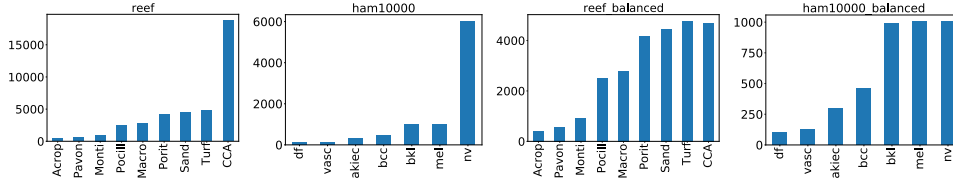


Figure A.7. Class distributions of MLC2008, HAM10000, and their balanced versions.

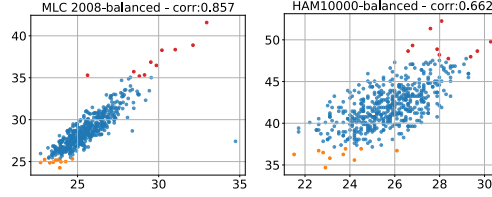


Figure A.8. Errors of all 500 sampled architectures on MLC2008-balanced and HAM10000-balanced (y-axis) plotted against the errors of their unbalanced counterparts (x-axis). The top 10 performances on the target dataset are plotted in orange, the worst 10 performances in red. We observe a clear positive correlation for both datasets, hence we conclude that the dataset imbalance has a limited impact on the APRs.

A.1.2 Additional ablation studies

A.1.2.1 Impact of Pretraining

The main objective of this study is to identify how well different CNN designs perform on varying datasets and if the best architectures are consistent across the datasets. For this reason we train all of our networks from scratch on each dataset. However, we cannot ignore that pretraining on ImageNet is a huge factor in practice and we therefore study its impact on our evaluations. To this end we have trained all of our sampled architectures again on each dataset but this time we initialize their weights with ImageNet pretraining (we omit Concrete, which has very low errors even without pretraining). Figure A.9 shows the errors of each dataset without (blue) and with (green) pretraining plotted against the ImageNet errors. The data shows a distinct trend: the overall performance improvement due to pretraining dictates how much stronger the ImageNet-correlation of the pretrained errors is compared to the errors without pretraining. For Cifar10 and Cifar100 where the performance gain with pretraining is low to moderate the error correlations do not drastically change. On the other end of the spectrum are Natural and Powerline, where pretraining leads to drastically lower errors.

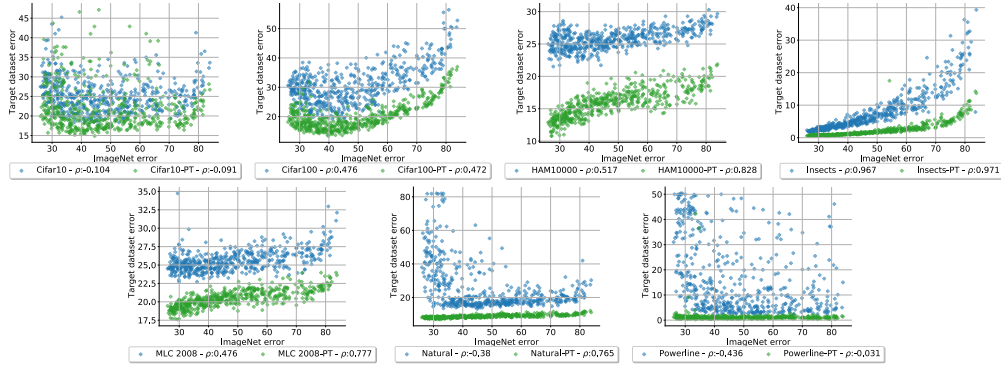


Figure A.9. Errors from all 500 architectures trained from scratch (blue) as well as the same architectures pretrained on ImageNet (green), plotted against the respective ImageNet errors. We observe that the error correlation with ImageNet increases relative to the performance gain due to pretraining.

This in turn leads to much higher error correlation with ImageNet (the Powerline correlation can not grow significantly above 0 because the overall errors are so small across all architectures). We can conclude that our findings are still valid when pretraining is used, but their effects can be masked when pretraining is the most important factor contributing to the overall final performance.

A.1.2.2 Structure of Top Performing Architectures

Figure A.10 shows the configuration of the top performing architecture in blue, as well as the mean and standard deviation of the top 15 configurations for every dataset. We observe that the top 15 architectures have very high variance in both bottleneck ratio and group width.

Block width on the other hand shows a clear pattern: almost all high-performing architectures start with a very small block width that increases across the stages. Only Powerline and Natural do not show this pattern. In block depth, we observe a similar pattern with a bit more noise. For block depth, Powerline, Natural, Cifar10 and Cifar100, no such trend of increased parameter values towards the later stages is observed. This reinforces the idea that block width and block depth greatly impact an architectures performance and their optimal choices are dataset dependent.

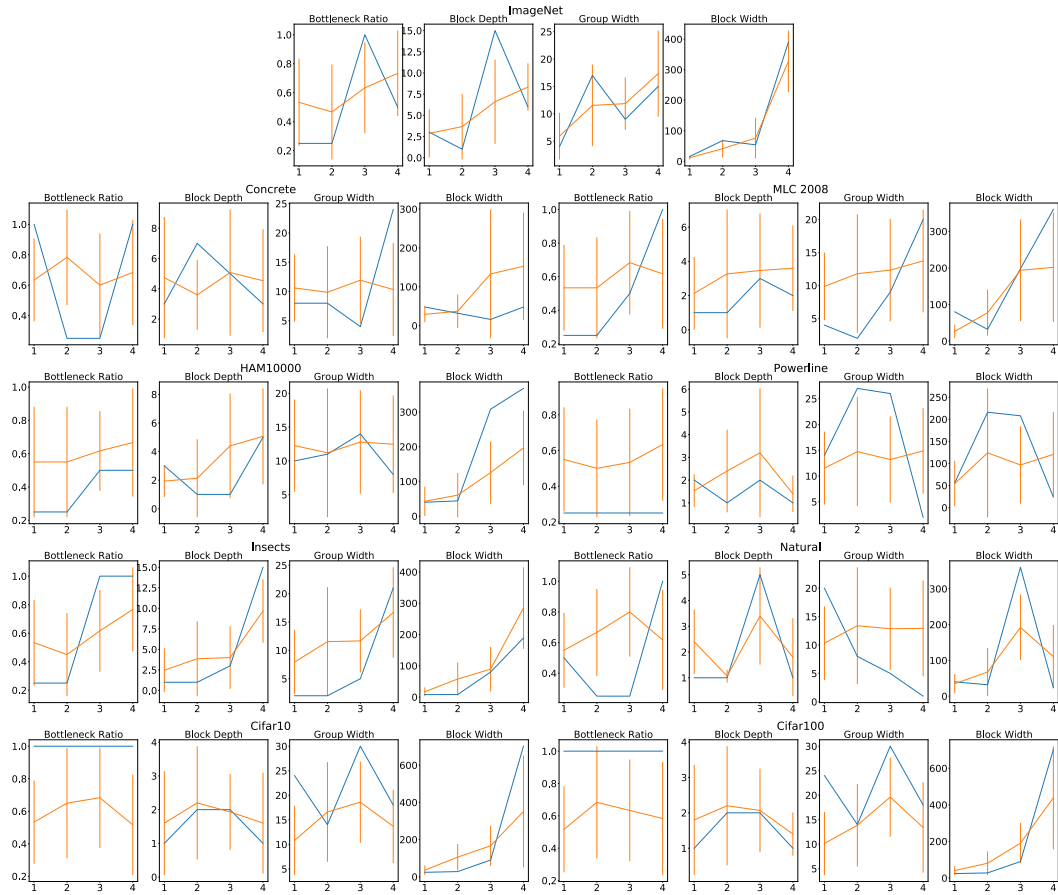


Figure A.10. Configurations of the top-performing architectures, with the four stages depicted on the x-axis and the parameter values on the y-axis. The best architectures are shown in blue, the mean of the top 15 architectures is depicted in orange with a vertical indication of one standard deviation.



Figure A.11. Matrix of error scatterplots of all datasets except Concrete (The first row replicates plots shown in Figure [7.8](#)).

A.2 Mental Rotation

A.2.1 Investigating on Stanford Cars and Oxford Pet

We investigate the phenomenon that for Stanford Cars (and to a lesser extent) Oxford Pet the AMR performance decreases when inputs are close to upright. We create polar plots of the Stanford Cars top-1 accuracies where AMR training duration and base architecture are shown individually (see Figure A.12). We observe that longer training of the AMR module helps to some extent. Using a small, data-efficient architecture such as ResNet18 on the other hand almost completely removes this effect. It shows that AMR module training is much more reliant on a sample-efficient, appropriately sized (with respect to the dataset) base model compared to standard classification training where large models tend to perform best. We, therefore, suspect that models which are oversized for a simple dataset such as Stanford Cars can learn very task-specific filters in early layers that only retain information used for classification. This in turn causes the angle classifier to fail when an upright image is used for which these filters have been optimized and very minimal information is retained in the features extracted from the base model.

A.2.2 Are the base network features useful for rotation estimation?

We opt to use the features generated by a base classification network as the input to our AMR module instead of using a distinct specialized network for angle prediction. This allows us to design an AMR module with a very low number of weights such that it is quickly trained. The intuition behind this choice is that the features which are trained for classification also carry valuable information for angle prediction. This should be especially true for the early layers which

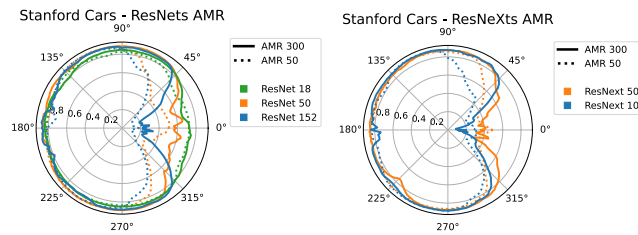


Figure A.12. Polar plots of Stanford Cars top-1 AMR accuracies, shown individually by training duration (dashed vs dotted) and base model architecture (color coded) instead of averaged.

Table A.2. Stanford Cars (top) and Oxford Pet (bottom) top-1 accuracies for all architectures analogous to the one shown above for ImageNet (see Table 8.1).

Stanford Cars									
Testing	Upright Training			Rotated Training		AMR 300		AMR 50	
	up	rot	% ceil	rot	% ceil	rot	% ceil	rot	% ceil
ResNet-18	0.854	0.163	0.19	0.412	0.48	0.812	0.95	0.800	0.94
ResNet-50	0.892	0.182	0.20	0.535	0.60	0.807	0.90	0.705	0.79
ResNet-152	0.886	0.169	0.19	0.671	0.76	0.737	0.83	0.615	0.69
EfficientNet-b0	0.825	0.140	0.17	0.687	0.83	0.820	0.99	0.785	0.95
EfficientNet-b2	0.831	0.138	0.17	0.770	0.93	0.823	0.99	0.808	0.97
EfficientNet-b4	0.885	0.163	0.18	0.786	0.89	0.877	0.99	0.860	0.97
ResNext-50-32x4d	0.877	0.185	0.21	0.507	0.58	0.763	0.87	0.728	0.83
ResNext-101-32x8d	0.885	0.179	0.20	0.577	0.65	0.725	0.82	0.666	0.75
Average	0.867	0.165	0.19	0.618	0.71	0.796	0.92	0.746	0.86

Oxford Pet									
Testing	Upright Training			Rotated Training		AMR 1000		AMR 150	
	up	rot	% ceil	rot	% ceil	rot	% ceil	rot	% ceil
ResNet-18	0.700	0.442	0.63	0.579	0.83	0.665	0.95	0.624	0.89
ResNet-50	0.750	0.477	0.64	0.590	0.79	0.712	0.95	0.660	0.88
ResNet-152	0.752	0.460	0.61	0.579	0.77	0.697	0.93	0.633	0.84
EfficientNet-b0	0.740	0.496	0.67	0.630	0.85	0.727	0.98	0.699	0.94
EfficientNet-b2	0.763	0.518	0.68	0.617	0.81	0.753	0.99	0.720	0.94
EfficientNet-b4	0.735	0.515	0.70	0.605	0.82	0.719	0.98	0.686	0.93
ResNext-50-32x4d	0.743	0.489	0.66	0.579	0.78	0.711	0.96	0.679	0.91
ResNext-101-32x8d	0.748	0.469	0.63	0.645	0.86	0.712	0.95	0.656	0.88
Average	0.741	0.483	0.65	0.603	0.81	0.712	0.96	0.670	0.90

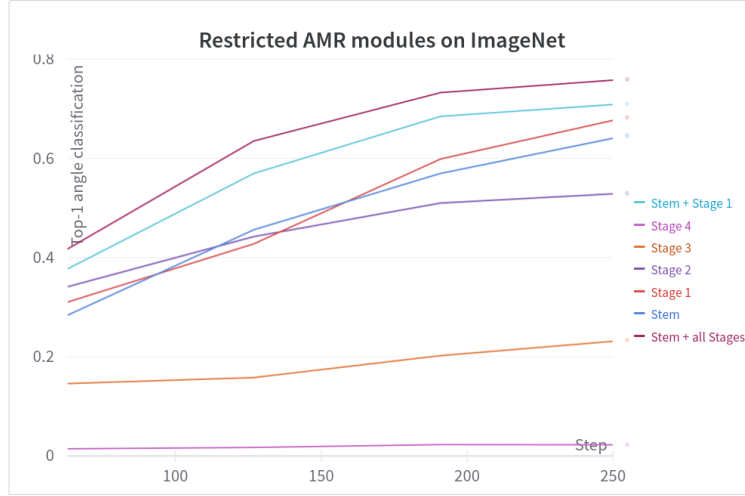


Figure A.13. Top-1 angle accuracies for ResNet-18+AMR with the information flow from the base network restricted. Legend indicates which channels (see Figure 2 in the main manuscript) from the base network are open.

tend to consist more of lower-level, class-agnostic, features. In this section, we present an ablation study exploring the validity of this assumption. We train AMR modules attached to a ResNet-18 with restricted information flow from the base model to the AMR. First, we train a module and only open the connection from the Stem to the AMR module (see Figure 8.2). We repeat this for each of the four other connections (Stage 1 - Stage 4). Figure A.13 shows the evolution of the top-1 angle classification errors over the training time. The best single source of information is the output of Stage 1. A close second is the module that is only attached to the stem (making it essentially a separate network). As expected are the outputs of the later stages far less informative. We train an additional module with input from Stage 1 plus Stem, it outperforms the other modules which are attached to only one of those quite significantly. This shows that the channels carry some complementary information. Finally, the module attached to the base network at all five connections clearly performs the best. Therefore we can conclude that the features of the base network are helpful for angle prediction and feeding the output of all base network stages into the AMR module is the best design choice.

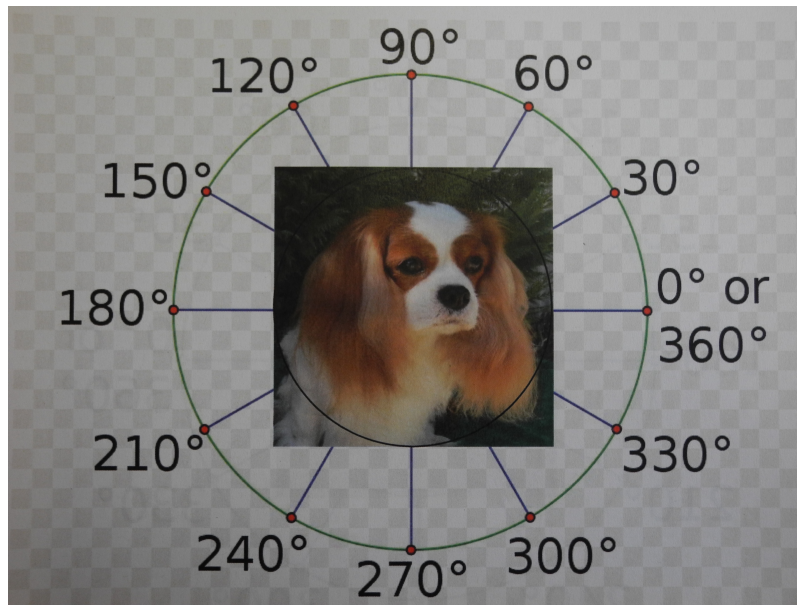


Figure A.14. Example printout featuring helper lines to facilitate photographs at exact angles.

A.2.3 Re-digitized Images

Figure [A.14](#) shows one example of the printouts that were used to create the manually rotated images for Section [8.5](#). The additional helper lines were printed to facilitate the creation of photographs of the printout at exactly the twelve desired angles. Figure [A.15](#) shows the re-digitized images at an angle (top rows) and after AMR (bottom rows) with their classifications color coded analogous to Figure [8.5](#) but for all seven test images.

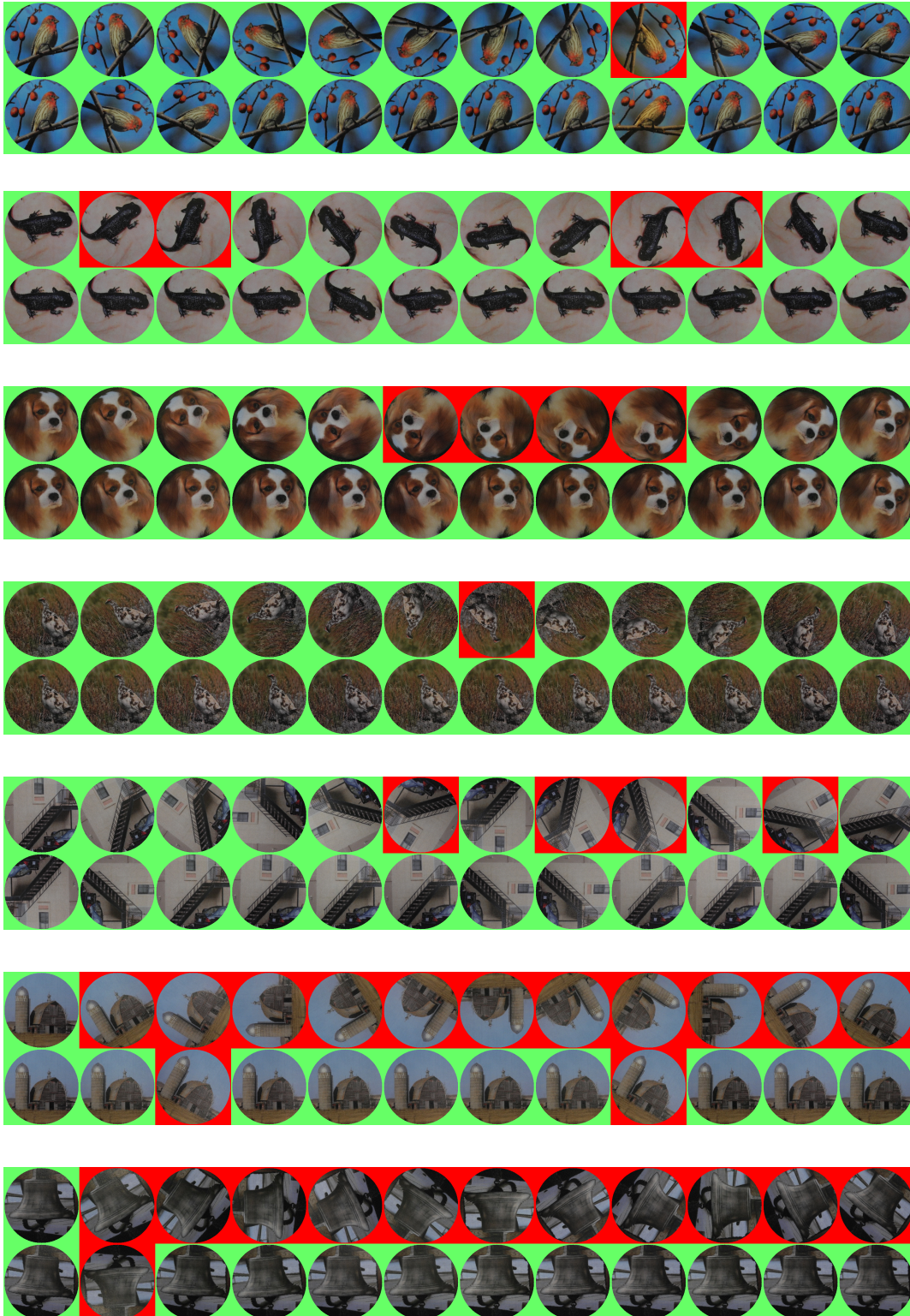


Figure A.15. All re-digitized photos raw (top rows) and after AMR (bottom rows) with their corresponding classifications color coded.

Appendix B

Artifacts

This Chapter provides links to the accompanying code, data, and trained models (where available) for this thesis.

B.1 DeepScoresV2 and RealScores

Our DeepScoresV2 artifacts (Datasets and Code) are curated and published on the open-data platform Zenodo: <https://zenodo.org/records/4012193>. The RealScores data and related code can be found here: https://github.com/raember/s2anet/tree/TISMIR_publication.

B.2 Is it enough to optimize CNN architectures on ImageNet?

Code, training configurations and training-run data can be found here: https://github.com/tuggeluk/pycls/tree/ImageNet_as_basis

B.3 Mental Rotation

Data

ImageNet, Stanford Cars, and Oxford Pet are public datasets and can be sourced from their original authors. Our photographed-at-an-angle versions of ImageNet images can be downloaded here: https://github.com/tuggeluk/ffcv-imagenet/tree/rotation_module/sanity_check_data.

Code

Our code is publicly available on GitHub at https://github.com/tuggeluk/ffcv-imagenet/tree/rotation_module.

Hyperparameters

We logged hyperparameters as well as relevant metrics of each run to wandb.ai. The easiest and safest way to exactly replicate a run is to first check out the git state and then use the run command given in the overview tab of each run. The logs of each training and evaluation run can be found under the URLs shown in Table [B.1](#).

Model weights

The model weights for each trained model we use here can be obtained using the following link https://github.com/tuggeluk/ffcv-imagenet/tree/rotation_module/downloads.md.

Table B.1. Links to wandb.ai projects containing hyperparameters and relevant metrics of all training and evaluation runs created for this work.

ImageNet CNN	
<i>Training</i>	
Base models	https://wandb.ai/tuggeluk/train_base_models
AMR modules	https://wandb.ai/tuggeluk/train_angleclass_no_lossshape
<i>Evaluation</i>	
Base models	https://wandb.ai/tuggeluk/evaluate_final_base_models_highres
BM + AMR	https://wandb.ai/tuggeluk/evaluate_final_angle_class_highres , https://wandb.ai/tuggeluk/evaluate_final_angle_class_highres_5ep
ImageNet ViT	
<i>Training</i>	
Base models	https://wandb.ai/tuggeluk/base%20ViT
AMR modules	https://wandb.ai/tuggeluk/train_angleclass_no_lossshapeViT
<i>Evaluation</i>	
Base models	https://wandb.ai/tuggeluk/evaluate_final_base_models_highres_ViT
BM + AMR	https://wandb.ai/tuggeluk/evaluate_angle_class_ViT
Stanford Cars	
<i>Training</i>	
Base models	https://wandb.ai/tuggeluk/train_base_models_stanfordcars
AMR modules	https://wandb.ai/tuggeluk/train_angleclass_no_lossshape_StanfordCars
<i>Evaluation</i>	
Base models	https://wandb.ai/tuggeluk/evaluate_final_base_models_highres_StanfordCars
BM + AMR	https://wandb.ai/tuggeluk/test_angleclass_stanfordcars
Oxford Pet	
<i>Training</i>	
Base models	https://wandb.ai/tuggeluk/train_base_models_oxfordpet
AMR modules	https://wandb.ai/tuggeluk/train_angleclass_no_lossshape_OxfordPet
<i>Evaluation</i>	
Base models	https://wandb.ai/tuggeluk/evaluate_final_base_models_highres_OxfordPet
BM + AMR	https://wandb.ai/tuggeluk/test_angleclass_oxfordpets
MNIST	
<i>Training</i>	
Base models	https://wandb.ai/tuggeluk/train_base_models_MNIST
AMR modules	https://wandb.ai/tuggeluk/train_angleclass_no_lossshape_MNIST

Bibliography

- Andrychowicz, M., Denil, M., Gomez, S., Hoffman, M. W., Pfau, D., Schaul, T., Shillingford, B. and De Freitas, N. [2016]. Learning to learn by gradient descent by gradient descent, *Advances in Neural Information Processing Systems*, pp. 3981–3989.
- Ash, T. [1989]. Dynamic node creation in backpropagation networks, *Connection science* **1**(4): 365–375.
- Bai, M. and Urtasun, R. [2017]. Deep watershed transform for instance segmentation, *IEEE Conference on Computer Vision and Pattern Recognition*.
- Bainbridge, D. and Bell, T. [2001]. The challenge of optical music recognition, *Computers and the Humanities* **35.2**: 95 – 121.
- Baird, H. S. [1992]. Document image defect models, *Structured Document Image Analysis* pp. 546–556.
- Bansal, P. [2018]. Intel image classification.
- Banzhaf, W., Nordin, P., Keller, R. E. and Francone, F. D. [1998]. *Genetic programming: an introduction*, Vol. 1.
- Baro, A., Riba, P. and Fornés, A. [2016]. Towards the recognition of compound music notes in handwritten music scores, *15th International Conference on Frontiers in Handwriting Recognition, ICFHR 2016, Shenzhen, China, October 23-26, 2016*, pp. 465–470.
- Baró-Mas, A. [2017]. *Optical music recognition by long short-term memory recurrent neural networks*, PhD thesis, Universitat Autònoma de Barcelona Bellaterra, Spain.
- Beijbom, O., Edmunds, P. J., Kline, D. I., Mitchell, B. G. and Kriegman, D. J. [2012]. Automated annotation of coral reef survey images, *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1170–1177.

- Bellini, P., Bruno, I. and Nesi, P. [2001]. Optical music sheet segmentation, *Proceedings First International Conference on WEB Delivering of Music. WEDELMUSIC 2001*, pp. 183–190.
- Bellini, P., Bruno, I. and Nesi, P. [2008]. Optical music recognition: Architecture and algorithms, *Interactive multimedia music technologies*, pp. 80–110.
- Beucher, S. et al. [1992]. The watershed transformation applied to image segmentation, *SCANNING MICROSCOPY-SUPPLEMENT* .
- Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E. et al. [2021]. On the opportunities and risks of foundation models, *arXiv preprint arXiv:2108.07258* .
- Boominathan, L., Srinivas, S. and Babu, R. V. [2016]. Compensating for large in-plane rotations in natural images, *Proceedings of the Tenth Indian Conference on Computer Vision, Graphics and Image Processing*, pp. 1–8.
- Bowman, S. R., Angeli, G., Potts, C. and Manning, C. D. [2015]. A large annotated corpus for learning natural language inference, *Conference on Empirical Methods in Natural Language Processing, EMNLP 2015*, pp. 632–642.
- Bradski, G. [2000]. The OpenCV Library, *Dr. Dobb's Journal of Software Tools* .
- Braschler, M., Stockinger, K. and Stadelmann (Eds.), T. [2019]. *Applied Data Science—Lessons Learned for the Data-Driven Business*.
- Brazdil, P. B. and Soares, C. [2000]. A comparison of ranking methods for classification algorithm selection, *European conference on machine learning*, pp. 63–75.
- Brochu, E., Cora, V. M. and De Freitas, N. [2010]. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning, *arXiv preprint arXiv:1012.2599* .
- Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Nee-lakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I. and Amodei, D. [2020].

- Language models are few-shot learners, in H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan and H. Lin (eds), *Annual Conference on Neural Information Processing Systems 2020, (NeurIPS)*.
- Calvo-Zaragoza, J., Jr, J. H. and Pacha, A. [2020]. Understanding optical music recognition, *ACM Computing Surveys (CSUR)* **53**(4): 1–35.
- Calvo-Zaragoza, J. and Oncina, J. [2014]. Recognition of pen-based music notation: The homus dataset, *International Conference on Pattern Recognition*, pp 3038-3043 .
- Calvo-Zaragoza, J., Rizo, D. and J.M., I. [2016]. Two (note) heads are better than one: pen-based multimodal interaction with music scores, *International Society of Music Information Retrieval conference* .
- Calvo-Zaragoza, J., Valero-Mas, J. J. and Pertusa, A. [2017]. End-to-end optical music recognition using neural networks, *Proceedings of the 18th International Society for Music Information Retrieval Conference*, pp. 472–477.
- Carter, N. P. [1989]. *Automatic recognition of printed music in the context of electronic publishing*.
- Chen, Y., Hoffman, M. W., Colmenarejo, S. G., Denil, M., Lillicrap, T. P., Botvinick, M. and de Freitas, N. [2017]. Learning to learn without gradient descent by gradient descent, *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 748–756.
- Chen, Y., Yang, T., Zhang, X., Meng, G., Xiao, X. and Sun, J. [2019]. Detnas: Backbone search for object detection, in H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. B. Fox and R. Garnett (eds), *32th Annual Conference on Neural Information Processing Systems*, pp. 6638–6648.
- Choi, K., Coüasnon, B., Ricquebourg, Y. and Zanibbi, R. [2017]. Bootstrapping samples of accidentals in dense piano scores for cnn-based detection, *12th International Workshop on Graphics Recognition, 14th IAPR International Conference on Document Analysis and Recognition, GREC@ICDAR 2017, Kyoto, Japan, November 9-15, 2017*, pp. 19–20.
- Chollet, F. [2017]. Xception: Deep learning with depthwise separable convolutions, *2017 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1800–1807.

- Choudhury, G. S., Droetboom, M., DiLauro, T., Fujinaga, I. and Harrington, B. [2000]. Optical music recognition system within a large-scale digitization project., *International Society for Music Information Retrieval Conference*.
- Ciresan, D. C., Giusti, A., Gambardella, L. M. and Schmidhuber, J. [2012]. Neural networks for segmenting neuronal structures in EM stacks, *ISBI Segmentation Challenge Competition: Abstracts*.
- Ciresan, D. C., Meier, U., Masci, J., Gambardella, L. M. and Schmidhuber, J. [2011a]. Flexible, high performance convolutional neural networks for image classification, *Twenty-second international joint conference on artificial intelligence*.
- Cireşan, D. C., Meier, U., Masci, J., Gambardella, L. M. and Schmidhuber, J. [2011b]. High-performance neural networks for visual object classification, *arXiv preprint arXiv:1102.0183*.
- Ciresan, D. C., Meier, U., Masci, J. and Schmidhuber, J. [2011]. A committee of neural networks for traffic sign classification, *International Joint Conference on Neural Networks (IJCNN)*, pp. 1918–1921.
- Cireşan, D., Meier, U., Masci, J. and Schmidhuber, J. [2012]. Multi-column deep neural network for traffic sign classification, *Neural networks* **32**: 333–338.
- Cireşan, D. C., Meier, U., Gambardella, L. M. and Schmidhuber, J. [2010]. Deep, big, simple neural nets for handwritten digit recognition, *Neural Computation* **22**(12): 3207–3220.
- Cohen, T. and Welling, M. [2016]. Group equivariant convolutional networks, *Proceedings of the 33rd International Conference on Machine Learning*, pp. 2990–2999.
- Collins, J., Sohl-Dickstein, J. and Sussillo, D. [2017]. Capacity and trainability in recurrent neural networks, *5th International Conference on Learning Representations*.
- Cortes, C. and Vapnik, V. [1995]. Support-vector networks, *Machine learning* **20**(3): 273–297.
- Cybenko, G. [1989]. Approximation by superpositions of a sigmoidal function, *Mathematics of control, signals and systems* **2**(4): 303–314.

- Deng, J., Dong, W., Socher, R., Li, L. J., Li, K. and Fei-Fei, L. [2009]. Imagenet: A large-scale hierarchical image database., *In Computer Vision and Pattern Recognition, IEEE Conference on* (pp. 248-255). *IEEE*. .
- Dieleman, S., Willett, K. W. and Dambre, J. [2015]. Rotation-invariant convolutional neural networks for galaxy morphology prediction, *Monthly notices of the royal astronomical society* **450**(2): 1441–1459.
- Ding, W. and Taylor, G. W. [2014]. " mental rotation" by optimizing transforming distance, *arXiv preprint arXiv:1406.3010* .
- Dinh, L., Pascanu, R., Bengio, S. and Bengio, Y. [2017]. Sharp minima can generalize for deep nets, *in* D. Precup and Y. W. Teh (eds), *34th International Conference on Machine Learning*, pp. 1019–1028.
- Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E. and Darrell, T. [2014]. Decaf: A deep convolutional activation feature for generic visual recognition, *31th International Conference on Machine Learning*, pp. 647–655.
- dos Santos Cardoso, J., Capela, A., Rebelo, A., Guedes, C. and da Costa, J. P. [2009]. Staff detection with stable paths, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **31**(6): 1134–1139.
- Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J. and Houlsby, N. [2021]. An image is worth 16x16 words: Transformers for image recognition at scale, *9th International Conference on Learning Representations, ICLR*.
- Droettboom, M., Fujinaga, I. and MacMillan, K. [2002]. Optical music interpretation, *Structural, Syntactic, and Statistical Pattern Recognition: Joint IAPR International Workshops SSPR 2002 and SPR 2002 Windsor, Ontario, Canada, August 6–9, 2002 Proceedings*, pp. 378–387.
- Eggenberger, K., Lindauer, M. and Hutter, F. [2018]. Neural networks for predicting algorithm runtime distributions., *IJCAI*, pp. 1442–1448.
- Elsken, T., Metzen, J.-H. and Hutter, F. [2018]. Simple and efficient architecture search for convolutional neural networks, *Proceedings of International Conference on Learning Representations (ICLR)*.
- Elsken, T., Metzen, J. H. and Hutter, F. [2019]. Neural architecture search: A survey, *Journal of Machine Learning Research* **20**.

- Engstrom, L., Tran, B., Tsipras, D., Schmidt, L. and Madry, A. [2019]. Exploring the landscape of spatial robustness, in K. Chaudhuri and R. Salakhutdinov (eds), *Proceedings of the 36th International Conference on Machine Learning*, Vol. 97 of *Proceedings of Machine Learning Research*, pp. 1802–1811.
- Everingham, M., Van Gool, L., Williams, C. K., Winn, J. and Zisserman, A. [2010]. The pascal visual object classes (voc) challenge., *International journal of computer vision*, 88(2), 303-338. .
- Fang, J., Zhou, D., Song, X., Jin, S., Yang, R. and Zhang, L. [2020]. Rotpredictor: Unsupervised canonical viewpoint learning for point cloud classification, *2020 International Conference on 3D Vision (3DV)*, pp. 987–996.
- Fei-Fei, L., Fergus, R. and Perona, P. [2004]. Learning generative visual models from few training examples: an incremental bayesian approach tested on 101 object categories., *In Computer Vision and Pattern Recognition, IEEE Conference on*. .
- Ferrand, M., Leite, J. A. and Cardoso, A. [1999]. Hypothetical reasoning: an application to optical music recognition., *Appia-Gulp-Prode*, Vol. 99, pp. 367–381.
- Feurer, M., Eggenberger, K., Falkner, S., Lindauer, M. and Hutter, F. [2018]. Practical automated machine learning for the automl challenge 2018, *International Workshop on AutoML at ICML*.
- Feurer, M. and Hutter, F. [2019]. Hyperparameter optimization, *Automated Machine Learning*, pp. 3–33.
- Feurer, M., Klein, A., Eggenberger, K., Springenberg, J., Blum, M. and Hutter, F. [2015]. Efficient and robust automated machine learning, *Advances in Neural Information Processing Systems*.
- Feurer, M., Springenberg, J. T. and Hutter, F. [2014]. Using meta-learning to initialize bayesian optimization of hyperparameters, *Proceedings of the 2014 International Conference on Meta-learning and Algorithm Selection-Volume 1201*, pp. 3–10.
- Figueiredo, E., Park, G., Farrar, C. R., Worden, K. and Figueiras, J. [2011]. Machine learning algorithms for damage detection under operational and environmental variability, *Structural Health Monitoring* 10(6): 559–572.

- Fornes, A., Dutta, A., Gordo, A. and Lladós, J. [2012]. A ground-truth of handwritten music score images for writer identification and staff removal, *International Journal on Document Analysis and Recognition*, Volume 15, Issue 3, pp 243-251 .
- Fornés, A., Lladós, J. and Sánchez, G. [2005]. Primitive segmentation in old handwritten music scores, *International Workshop on Graphics Recognition*, pp. 279–290.
- Freedman, D., Pisani, R. and Purves, R. [2007]. Statistics (international student edition), *Pisani, R. Purves, 4th edn. WW Norton & Company, New York* .
- Fujinaga, I. [2004]. Staff detection and removal, *Visual Perception of Music Notation: On-Line and Off Line Recognition*, pp. 1–39.
- Fukushima, K. [1980]. Neocognitron: A self-organizing neural network for a mechanism of pattern recognition unaffected by shift in position, *Biological Cybernetics* **36**(4): 193–202.
- Fukushima, K. and Miyake, S. [1982]. Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position, *Pattern Recognition* **15**(6): 455–469.
- Fukushima, K., Miyake, S. and Ito, T. [1983]. Neocognitron: A neural network model for a mechanism of visual pattern recognition, *IEEE transactions on systems, man, and cybernetics* (5): 826–834.
- Gaudel, R. and Sebag, M. [2010]. Feature selection as a one-player game, *International Conference on Machine Learning*, pp. 359–366.
- Geman, S., Bienenstock, E. and Doursat, R. [1992]. Neural networks and the bias/variance dilemma, *Neural computation* **4**(1): 1–58.
- Girshick, R. [2015]. Fast R-CNN, *Proceedings of the IEEE international conference on computer vision*, pp. 1440–1448.
- Girshick, R. B., Donahue, J., Darrell, T. and Malik, J. [2014]. Rich feature hierarchies for accurate object detection and semantic segmentation, *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 580–587.
- Girshick, R., Radosavovic, I., Gkioxari, G., Dollár, P. and He, K. [2018]. Detectron, <https://github.com/facebookresearch/detectron>.

- Göcke, R. [2003]. Building a system for writer identification on handwritten music scores, *Proceedings of the IASTED International Conference on Signal Processing, Pattern Recognition, and Applications (SPPRA)*, pp. 250–255.
- Goodfellow, I. J., Bulatov, Y., Ibarz, J., Arnoud, S. and Shet, V. [2013]. Multi-digit number recognition from street view imagery using deep convolutional neural networks., *arXiv preprint arXiv:1312.6082*. .
- Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R. and Schmidhuber, J. [2017]. LSTM: A search space odyssey, *IEEE Trans. Neural Networks Learn. Syst.* **28**(10): 2222–2232.
- Gregory, G., Alex, H. and Pietro, P [2007]. Caltech-256 object category dataset, *Technical Report - California Institute of Technology* .
- Guyon, I., Sun-Hosoya, L., Boullé, M., Escalante, H., Escalera, S., Liu, Z., Jajetic, D., Ray, B., Saeed, M., Sebag, M. et al. [2017]. Analysis of the automl challenge series 2015-2018.
- Hajic Jr, J., Dorfer, M., Widmer, G. and Pecina, P [2018]. Towards full-pipeline handwritten omr with musical symbol detection by u-nets., *International Society for Music Information Retrieval Conference*, pp. 225–232.
- Handzic, M., Tjandrawibawa, F and Yeo, J. [2003]. How neural networks can help loan officers to make better informed application decisions, *Informing Science* **6**: 97–109.
- Hansen, O. L. P., Svenning, J.-C., Olsen, K., Dupont, S., Garner, B. H., Iosifidis, A., Price, B. W. and Høye, T. T. [2019]. Image data used for publication "Species-level image classification with convolutional neural network enable insect identification from habitus images ".
- Hanson, S. and Pratt, L. [1988]. Comparing biases for minimal network construction with back-propagation, *Advances in neural information processing systems* **1**.
- He, K., Gkioxari, G., Dollar, P and Girshick, R. [2017]. Mask r-cnn., *In Proceedings of the IEEE international conference of computer vision* .
- He, K., Zhang, X., Ren, S. and Sun, J. [2016]. Deep residual learning for image recognition, *2016 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778.

- He, Y., Lin, J., Liu, Z., Wang, H., Li, L.-J. and Han, S. [2018]. Amc: Automl for model compression and acceleration on mobile devices, *Proceedings of the European Conference on Computer Vision*, pp. 784–800.
- Hestness, J., Narang, S., Ardalani, N., Diamos, G. F., Jun, H., Kianinejad, H., Patwary, M. M. A., Yang, Y. and Zhou, Y. [2017]. Deep learning scaling is predictable, empirically, *arXiv preprint arXiv:1712.00409* .
- Hochreiter, S. and Schmidhuber, J. [1997]. Long short-term memory, *Neural computation* **9**(8): 1735–1780.
- Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M. and Adam, H. [2017]. Mobilenets: Efficient convolutional neural networks for mobile vision applications, *arXiv preprint arXiv:1704.04861* .
- Hu, J., Shen, L. and Sun, G. [2018]. Squeeze-and-excitation networks, *2018 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7132–7141.
- Hutter, F., Hoos, H. H. and Leyton-Brown, K. [2011]. Sequential model-based optimization for general algorithm configuration, *International Conference on Learning and Intelligent Optimization*, pp. 507–523.
- Hutter, F., Kotthoff, L. and Vanschoren, J. [2019]. Automatic machine learning: methods, systems, challenges, *Challenges in Machine Learning* .
- Ivakhnenko, A. G. [1968]. The group method of data handling; a rival of the method of stochastic approximation.
- Jaderberg, M., Dalibard, V., Osindero, S., Czarnecki, W. M., Donahue, J., Razavi, A., Vinyals, O., Green, T., Dunning, I., Simonyan, K. et al. [2017]. Population based training of neural networks, *arXiv preprint arXiv:1711.09846* .
- Jamieson, K. and Talwalkar, A. [2016]. Non-stochastic best arm identification and hyperparameter optimization, *Artificial Intelligence and Statistics*, pp. 240–248.
- Jan Hajič, j. and Pecina, P. [2017]. The MUSCIMA++ Dataset for Handwritten Optical Music Recognition, *14th International Conference on Document Analysis and Recognition, ICDAR 2017, Kyoto, Japan, November 13 - 15, 2017*, New York, USA, pp. 39–46.
- Jing, L. and Tian, Y. [2019]. Self-supervised visual feature learning with deep neural networks: A survey, *arXiv preprint arXiv:1902.06162* .

- Jr., J. H. and Pecina, P. [2017]. Detecting noteheads in handwritten scores with convnets and bounding box regression, *arXiv preprint arXiv:1708.01806* .
- Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A. et al. [2021]. Highly accurate protein structure prediction with alphafold, *nature* **596**(7873): 583–589.
- Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J. and Amodei, D. [2020a]. Scaling laws for neural language models, *CoRR* **abs/2001.08361**.
URL: <https://arxiv.org/abs/2001.08361>
- Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J. and Amodei, D. [2020b]. Scaling laws for neural language models, *arXiv preprint arXiv:2001.08361* .
- Katz, G., Shin, E. C. R. and Song, D. [2016]. Explorekit: Automatic feature generation and selection, *Data Mining (ICDM), 2016 IEEE 16th International Conference on*, pp. 979–984.
- Kaul, A., Maheshwary, S. and Pudi, V. [2017]. Autolearn—automated feature generation and selection, *Data Mining (ICDM), 2017 IEEE International Conference on*, pp. 217–226.
- Kawaguchi, K., Kaelbling, L. P. and Bengio, Y. [2017]. Generalization in deep learning, *arXiv preprint arXiv:1710.05468* .
- Keurti, H., Pan, H., Besserve, M., Grewe, B. F. and Schölkopf, B. [2023]. Homomorphism autoencoder - learning group structured representations from observed transitions, in A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato and J. Scarlett (eds), *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, Proceedings of Machine Learning Research, PMLR, pp. 16190–16215.
- Kingma, D. and Ba, J. [2015]. Adam: A method for stochastic optimization, *International Conference on Learning Representations (ICLR)* .
- Kitano, H. [1990]. Designing neural networks using genetic algorithms with graph generation system, *Complex System* **4**(4): 461–476.
- Klein, A., Falkner, S., Springenberg, J. T. and Hutter, F. [2016]. Learning curve prediction with bayesian neural networks.

- Kornblith, S., Shlens, J. and Le, Q. V. [2019]. Do better imagenet models transfer better?, *2019 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2661–2671.
- Krause, J., Stark, M., Deng, J. and Fei-Fei, L. [2013]. 3d object representations for fine-grained categorization, *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, pp. 554–561.
- Krizhevsky, A., Hinton, G. et al. [2009]. Learning multiple layers of features from tiny images.
- Krizhevsky, A., Sutskever, I. and Hinton, G. E. [2012]. Imagenet classification with deep convolutional neural networks., *In Advances in neural information processing systems (pp. 1097-1105)*. .
- Laptev, D., Savinov, N., Buhmann, J. M. and Pollefeys, M. [2016]. Ti-pooling: transformation-invariant pooling for feature learning in convolutional neural networks, *IEEE conference on computer vision and pattern recognition*, pp. 289–297.
- Leclerc, G., Ilyas, A., Engstrom, L., Park, S. M., Salman, H. and Madry, A. [2022]. FFCV: Accelerating training by removing data bottlenecks, <https://github.com/Libffcv/ffcv/>.
- LeCun, Y., Boser, B. E., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. E. and Jackel, L. D. [1989]. Backpropagation applied to handwritten zip code recognition, *Neural Computation* .
- LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P. [1998]. Gradient-based learning applied to document recognition, *Proceedings of the IEEE* **86**(11): 2278–2324.
- LeCun, Y., Cortes, C. and Burges, C. J. [1998]. The mnist database of handwritten digits.
- LeCun, Y., Denker, J. and Solla, S. [1989]. Optimal brain damage, *Advances in neural information processing systems* **2**.
- Lee, C. and Osindero, S. [2016]. Recursive recurrent nets with attention modeling for OCR in the wild, *Conference on Computer Vision and Pattern Recognition*, pp. 2231–2239.
- Lee, H. C. [2019]. Autocv 2nd place solution. <https://github.com/HeungChangLee/autocv>.

- Li, H., Parikh, D., He, Q., Qian, B., Li, Z., Fang, D. and Hampapur, A. [2014]. Improving rail network velocity: A machine learning approach to predictive maintenance, *Transportation Research Part C: Emerging Technologies* **45**: 17–26.
- Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A. and Talwalkar, A. [2017]. Hyperband: A novel bandit-based approach to hyperparameter optimization, *The Journal of Machine Learning Research* **18**(1): 6765–6816.
- Lim, S., Kim, I., Kim, T., Kim, C. and Kim, S. [2019]. Fast autoaugment, *Advances in Neural Information Processing Systems*, pp. 6665–6675.
- Lin, G., Milan, A., Shen, C. and Reid, I. D. [2017]. Refinenet: Multi-path refinement networks for high-resolution semantic segmentation, *IEEE Conference on Computer Vision and Pattern Recognition*.
- Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D. and Zitnick, C. L. [2014]. Microsoft coco: Common objects in context., *In European conference on computer vision (pp. 740-755)*. Springer, Cham. .
- Linnainmaa, S. [1970]. The representation of the cumulative rounding error of an algorithm as a taylor expansion of the local rounding errors, *Master's Thesis (in Finnish)*, Univ. Helsinki pp. 6–7.
- Liu, C.-L., Yin, F., Wang, D.-H. and Wang, Q.-F. [2011]. Casia online and offline chinese handwriting databases, *Proc. 11th ICDAR*.
- Liu, H., Simonyan, K. and Yang, Y. [2019]. DARTS: differentiable architecture search, *7th International Conference on Learning Representations*.
- Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S. E., Fu, C. and Berg, A. C. [2016a]. SSD: single shot multibox detector, *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part I*, pp. 21–37.
- Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y. and Berg, A. C. [2016b]. Ssd: Single shot multibox detector., *In European conference on computer vision (pp. 21-37)*. Springer, Cham. .
- Liu, Z. [2018]. Autocv/ autodl starting kit. https://github.com/zhengying-liu/autodl_starting_kit_stable.

- Liu, Z., Guyon, I., Junior, J. J., Madadi, M., Escalera, S., Pavao, A., Escalante, H. J., Tu, W.-W., Xu, Z. and Treguer, S. [2019]. AutoCV Challenge Design and Baseline Results, *CAP 2019 - Conférence sur l'Apprentissage Automatique*, Toulouse, France.
- Long, J., Shelhamer, E. and Darrell, T. [2015]. Fully convolutional networks for semantic segmentation., *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 3431-3440). .
- Ma, Y., Tsao, D. and Shum, H.-Y. [2022]. On the principles of parsimony and self-consistency for the emergence of intelligence, *Frontiers of Information Technology & Electronic Engineering* **23**.
- MacKay, D. J. [1992]. A practical bayesian framework for backpropagation networks, *Neural computation* **4**(3): 448–472.
- Maclaurin, D., Duvenaud, D. and Adams, R. [2015]. Gradient-based hyperparameter optimization through reversible learning, *International Conference on Machine Learning*, pp. 2113–2122.
- maintainers, T. and contributors [2016]. Torchvision: Pytorch's computer vision library, <https://github.com/pytorch/vision>.
- Marcos, D., Volpi, M., Komodakis, N. and Tuia, D. [2017]. Rotation equivariant vector field networks, *IEEE International Conference on Computer Vision, ICCV*, pp. 5058–5067.
- Meier, B. B., Elezi, I., Amirian, M., Dürr, O. and Stadelmann, T. [2018]. Learning neural models for end-to-end clustering, *IAPR Workshop on Artificial Neural Networks in Pattern Recognition*, pp. 126–138.
- Meinshausen, N. and Bühlmann, P. [2010]. Stability selection, *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **72**(4): 417–473.
- Melis, G., Dyer, C. and Blunsom, P. [2018]. On the state of the art of evaluation in neural language models, *6th International Conference on Learning Representations*,.
- MIYAO, H. and NAKANO, Y. [1996]. Note symbol extraction for printed piano scores using neural networks, *IEICE TRANSACTIONS on Information and Systems* **79**(5): 548–554.

- Moody, J. [1991]. The effective number of parameters: An analysis of generalization and regularization in nonlinear learning systems, *Advances in neural information processing systems* 4.
- Nargesian, F., Samulowitz, H., Khurana, U., Khalil, E. B. and Turaga, D. [2017]. Learning feature engineering for classification, *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI*, Vol. 17, pp. 2529–2535.
- Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B. and Ng, A. Y. [2011]. Reading digits in natural images with unsupervised feature learning., *In NIPS workshop on deep learning and unsupervised feature learning (Vol. 2011, No. 2, p. 5)* .
- Nikolenko, S. I. et al. [2021]. *Synthetic data for deep learning*.
- Novak, R., Bahri, Y., Abolafia, D. A., Pennington, J. and Sohl-Dickstein, J. [2018]. Sensitivity and generalization in neural networks: an empirical study, *6th International Conference on Learning Representations*.
- Oh, K.-S. and Jung, K. [2004]. GPU implementation of neural networks, *Pattern Recognition* 37(6): 1311–1314.
- Olson, R. S., Urbanowicz, R. J., Andrews, P. C., Lavender, N. A., Moore, J. H. et al. [2016]. Automating biomedical data science through tree-based pipeline optimization, *European Conference on the Applications of Evolutionary Computation*, pp. 123–137.
- Özgenel, Ç. F. and Sorguç, A. G. [2018]. Performance comparison of pretrained convolutional neural networks on crack detection in buildings, *ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction*, pp. 1–8.
- Pacha, A. and Calvo-Zaragoza, J. [2018]. Optical music recognition in mensural notation with region-based convolutional neural networks., *International Society for Music Information Retrieval Conference*, pp. 240–247.
- Pacha, A., Calvo-Zaragoza, J. and Hajic Jr, J. [2019]. Learning notation graph construction for full-pipeline optical music recognition., *International Society for Music Information Retrieval Conference*, pp. 75–82.
- Pacha, A., Choi, K., Coüasnon, B., Ricquebourg, Y. and Zanibbi, R. [2018]. Handwritten music object detection: Open issues and baseline results, *International Workshop on Document Analysis Systems*.

- Pacha, A. and Eidenberger, H. [2017]. Towards self-learning optical music recognition, *16th IEEE International Conference on Machine Learning and Applications*, pp. 795–800.
- Pacha, A., Hajič jr., J. and Calvo-Zaragoza, J. [2018]. A baseline for general music object detection with deep learning, *Applied Sciences* **8**(9): 1488–1508.
- Parkhi, O. M., Vedaldi, A., Zisserman, A. and Jawahar, C. V. [2012]. Cats and dogs, *IEEE Conference on Computer Vision and Pattern Recognition*.
- Pérez, J. M., Muguerza, J., Arbelaitz, O., Gurrutxaga, I. and Martín, J. I. [2005]. Consolidated tree classifier learning in a car insurance fraud detection domain with class imbalance, *International Conference on Pattern Recognition and Image Analysis*, pp. 381–389.
- Pfahringer, B., Bensusan, H. and Giraud-Carrier, C. G. [2000]. Meta-learning by landmarking various learning algorithms., *ICML*, pp. 743–750.
- Poggio, T. A., Liao, Q., Miranda, B., Banburski, A., Boix, X. and Hidary, J. [2018]. Theory iiib: Generalization in deep networks, *CoRR* **abs/1806.11379**.
URL: <http://arxiv.org/abs/1806.11379>
- Prearu, D. [1970]. Computer pattern recognition of standard engraved music notation, *Ph. D thesis, Massachusetts Institute of Technology* .
- Quiroga, F., Ronchetti, F., Lanzarini, L. and Bariviera, A. F. [2020]. Revisiting data augmentation for rotational invariance in convolutional neural networks, *Modelling and Simulation in Management Sciences: Proceedings of the International Conference on Modelling and Simulation in Management Sciences (MS-18)*, pp. 127–141.
- Radosavovic, I., Johnson, J., Xie, S., Lo, W. and Dollár, P. [2019]. On network design spaces for visual recognition, *International Conference on Computer Vision*, pp. 1882–1890.
- Radosavovic, I., Kosaraju, R. P., Girshick, R. B., He, K. and Dollár, P. [2020]. Designing network design spaces, *2020 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 10425–10433.
- Raina, R., Madhavan, A. and Ng, A. Y. [2009]. Large-scale deep unsupervised learning using graphics processors, *In Proceedings of the 26th annual international conference on machine learning* (pp. 873-880) .

- Ramachandran, P., Zoph, B. and Le, Q. V. [2018]. Searching for activation functions, *6th International Conference on Learning Representations Workshop Track Proceedings*.
- Randriamahefa, R., Cocquerez, J. P., Fluhr, C., Pepin, F. and Philipp, S. [1993]. Printed music recognition, *Proceedings of 2nd International Conference on Document Analysis and Recognition (ICDAR'93)*, pp. 898–901.
- Razavian, A. S., Azizpour, H., Sullivan, J. and Carlsson, S. [2014]. CNN features off-the-shelf: An astounding baseline for recognition, *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 512–519.
- Real, E., Aggarwal, A., Huang, Y. and Le, Q. V. [2019]. Regularized evolution for image classifier architecture search, *The Thirty-Third AAAI Conference on Artificial Intelligence*, pp. 4780–4789.
- Real, E., Moore, S., Selle, A., Saxena, S., Suematsu, Y. L., Tan, J., Le, Q. V. and Kurakin, A. [2017]. Large-scale evolution of image classifiers, in D. Precup and Y. W. Teh (eds), *Proceedings of the 34th International Conference on Machine Learning*, Vol. 70 of *Proceedings of Machine Learning Research*, International Convention Centre, Sydney, Australia, pp. 2902–2911.
- Rebelo, A., Capela, G. and Cardoso, J. S. [2010]. Optical recognition of music symbols - A comparative study, *IJDAR* **13**(1): 19–31.
- Rebelo, A., Fujinaga, I., Paszkiewicz, F., Marcal, A. R. S., Guedes, C. and Cardoso, J. S. [2012]. Optical music recognition: state-of-the-art and open issues, *International Journal of Multimedia Information Retrieval* **1**(3): 173–190.
- Recht, B., Roelofs, R., Schmidt, L. and Shankar, V. [2019]. Do imagenet classifiers generalize to imagenet?, in K. Chaudhuri and R. Salakhutdinov (eds), *36th International Conference on Machine Learning*, pp. 5389–5400.
- Redmon, J., Divvala, S., Girshick, R. and Farhadi, A. [2016]. You only look once: Unified, real-time object detection., In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 779-788). .
- Redmon, J. and Farhadi, A. [2017]. YOLO9000: better, faster, stronger, *2017 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6517–6525.
- Reed, K. T. and Parker, J. R. [1996]. Automatic computer recognition of printed music, *Proceedings of 13th International Conference on Pattern Recognition*, Vol. 3, pp. 803–807.

- Ren, S., He, K., Girshick, R. and Sun, J. [2014]. Faster r-cnn: Towards real-time object detection with region proposal networks., *In Advances in neural information processing systems* (pp. 91-99). .
- Roach, J. W. and Tatem, J. [1988]. Using domain knowledge in low-level visual processing to interpret handwritten music: an experiment, *Pattern recognition* **21**(1): 33–44.
- Romero, D. W. and Cordonnier, J. [2021]. Group equivariant stand-alone self-attention for vision, *9th International Conference on Learning Representations, ICLR*.
- Rosenfeld, J. S., Rosenfeld, A., Belinkov, Y. and Shavit, N. [2020]. A constructive prediction of the generalization error across scales, *8th International Conference on Learning Representations*.
- Rossant, F. and Bloch, I. [2007]. Robust and adaptive OMR system including fuzzy modeling, fusion of musical rules, and possible error detection, *EURASIP J. Adv. Sig. Proc.* **2007**.
- Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C. and Fei-Fei, L. [2015]. ImageNet Large Scale Visual Recognition Challenge, *International Journal of Computer Vision (IJCV)* **115**(3): 211–252.
- Sánchez, A. J. G. and Calvo-Zaragoza, J. [2017]. Staff-line removal with selectional auto-encoders, *Expert Syst. Appl.* **89**: 138–148.
- Sandler, M., Howard, A. G., Zhu, M., Zhmoginov, A. and Chen, L. [2018]. MobileNetV2: Inverted residuals and linear bottlenecks, *2018 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4510–4520.
- Sato, I., Nishimura, H. and Yokoi, K. [2015]. Apac: Augmented pattern classification with neural networks, *arXiv preprint arXiv:1505.03229* .
- Schlag, I., Irie, K. and Schmidhuber, J. [2021]. Linear transformers are secretly fast weight programmers, *International Conference on Machine Learning*, pp. 9355–9366.
- Schmidhuber, J. [1992]. Learning to control fast-weight memories: An alternative to dynamic recurrent networks, *Neural Comput.* **4**(1): 131–139.

- Schmidhuber, J. [1997]. Discovering neural nets with low kolmogorov complexity and high generalization capability, *Neural Networks* **10**(5): 857–873.
- Schmidhuber, J. [2015]. Deep learning in neural networks: An overview, *Neural Networks* **61**: 85–117.
- Schölkopf, B. [2001]. The kernel trick for distances, *Advances in neural information processing systems*, pp. 301–307.
- Shafahi, A., Najibi, M., Ghiasi, M. A., Xu, Z., Dickerson, J., Studer, C., Davis, L. S., Taylor, G. and Goldstein, T. [2019]. Adversarial training for free!, *Advances in Neural Information Processing Systems* 32, pp. 3353–3364.
- Shepard, R. N. and Metzler, J. [1971]. Mental rotation of three-dimensional objects, *Science* **171**(3972): 701–703.
- Shihavuddin, A. S. M., Gracias, N., García, R., Gleason, A. C. R. and Gintert, B. [2013]. Image-based coral reef classification and thematic mapping, *Remote Sens.* **5**(4): 1809–1841.
- Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A. et al. [2017]. Mastering the game of go without human knowledge, *nature* **550**(7676): 354–359.
- Simard, P. Y., Steinkraus, D. and Platt, J. C. [2003]. Best practices for convolutional neural networks applied to visual document analysis, *7th International Conference on Document Analysis and Recognition (ICDAR)*, pp. 958–962.
- Simonyan, K. and Zisserman, A. [2015]. Very deep convolutional networks for large-scale image recognition, *3rd International Conference on Learning Representations*.
- Spreadbury, D. and Piéchaud, R. [2015]. Standard music font layout (SMuFL), *First International Conference on Technologies for Music Notation and Representation - TENOR2015*, Paris, France, pp. 146–153.
- Srivastava, R. K., Greff, K. and Schmidhuber, J. [2015a]. Highway networks, *arXiv preprint arXiv:1505.00387*.
- Srivastava, R. K., Greff, K. and Schmidhuber, J. [2015b]. Training very deep networks, *Advances in Neural Information Processing Systems (NIPS)*.

- Stadelmann, T., Amirian, M., Arabaci, I., Arnold, M., Duivesteijn, G. F., Elezi, I., Geiger, M., Lörwald, S., Meier, B. B., Rombach, K. et al. [2018]. Deep learning in the wild, *IAPR Workshop on Artificial Neural Networks in Pattern Recognition*, pp. 17–38.
- Stadelmann, T., Tolkachev, V., Sick, B., Stampfli, J. and Dürr, O. [2019]. Beyond imagenet: deep learning in industrial practice, *Applied Data Science*, pp. 205–232.
- Stallkamp, J., Schlipsing, M., Salmen, J. and Igel, C. [2011]. The german traffic sign recognition benchmark: a multi-class classification competition., *Proc. 11th ICDAR* .
- Stühler, E., Braune, S., Lionetto, F., Heer, Y., Kassraian-Fard, P., Jules, E., Westermann, C., Bergmann, A., van Hövell, P. and Group, N. S. [submitted]. Framework for personalized prediction of treatment response in relapsing remitting multiple sclerosis, *BMC medical research methodology* .
- Sun, K., Xiao, B., Liu, D. and Wang, J. [2019]. Deep high-resolution representation learning for human pose estimation, *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5693–5703.
- Susto, G. A., Schirru, A., Pampuri, S., McLoone, S. and Beghi, A. [2015]. Machine learning for predictive maintenance: A multiple classifier approach, *IEEE Transactions on Industrial Informatics* **11**(3): 812–820.
- Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Vanhoucke, V. and Rabinovich, A. [2015]. Going deeper with convolutions., *In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1-9)*. .
- Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. and Fergus, R. [2013]. Intriguing properties of neural networks., *arXiv:1312.6199*. .
- Tan, M. and Le, Q. V. [2019]. Efficientnet: Rethinking model scaling for convolutional neural networks, *in K. Chaudhuri and R. Salakhutdinov (eds), 36th International Conference on Machine Learning*, pp. 6105–6114.
- Thornton, C., Hutter, F., Hoos, H. H. and Leyton-Brown, K. [2013]. Auto-weka: Combined selection and hyperparameter optimization of classification algorithms, *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 847–855.

- Tieleman, T. and Hinton, G. E. [2012]. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude, *COURSERA: Neural networks for machine learning* 4.2 pp. 26–31.
- Torralba, A. and Efros, A. A. [2011]. Unbiased look at dataset bias, *Computer Vision and Pattern Recognition (pp. 1521-1528). IEEE*. .
- Torralba, A., Fergus, R. and Freeman, W. T. [2008]. 80 million tiny images: A large data set for nonparametric object and scene recognition, *IEEE Trans. Pattern Anal. Mach. Intell.* **30**(11): 1958–1970.
- Toyama, F., Shoji, K. and Miyamichi, J. [2006]. Symbol recognition of printed piano scores with touching symbols, *18th International Conference on Pattern Recognition (ICPR'06)*, Vol. 2, pp. 480–483.
- Tschandl, P., Rosendahl, C. and Kittler, H. [2018]. The HAM10000 dataset: A large collection of multi-source dermatoscopic images of common pigmented skin lesions, *arXiv preprint arXiv:1803.10417* .
- Tsoumakas, G. [2018]. A survey of machine learning techniques for food sales prediction, *Artificial Intelligence Review* pp. 1–7.
- Tuggener, L., Amirian, M., Benites, F., von Däniken, P., Gupta, P., Schilling, F-P and Stadelmann, T. [2020]. Design patterns for resource-constrained automated deep-learning methods, *AI* **1**(4): 510–538.
- Tuggener, L., Amirian, M., Rombach, K., Lörwald, S., Varlet, A., Westermann, C. and Stadelmann, T. [2019]. Automated machine learning in practice: state of the art and recent results, *2019 6th Swiss Conference on Data Science (SDS)*, pp. 31–36.
- Tuggener, L., Elezi, I., Schmidhuber, J., Pelillo, M. and Stadelmann, T. [2018]. Deepscores-a dataset for segmentation, detection and classification of tiny objects, *2018 24th International Conference on Pattern Recognition (ICPR)*, pp. 3704–3709.
- Tuggener, L., Elezi, I., Schmidhuber, J. and Stadelmann, T. [2018]. Deep watershed detector for music object recognition, *2018 In 19th International Society for Music Information Retrieval Conference*, pp. 23–27.
- Tuggener, L., Emberger, R., Ghosh, A., Sager, P., Satyawar, Y. P., Montoya, J., Goldschagg, S., Seibold, F., Gut, U., Ackermann, P. et al. [2023]. Real world

- music object recognition, *Transactions of the International Society for Music Information Retrieval* .
- Tuggener, L., Satyawar, Y. P., Pacha, A., Schmidhuber, J. and Stadelmann, T. [2021]. The deepscoresv2 dataset and benchmark for music object detection, *2020 25th International Conference on Pattern Recognition (ICPR)*, pp. 9188–9195.
- Tuggener, L., Schmidhuber, J. and Stadelmann, T. [2022]. Is it enough to optimize cnn architectures on imagenet?, *Frontiers in Computer Science* **4**.
- Tuggener, L., Stadelmann, T. and Schmidhuber, J. [2024]. Efficient rotation invariance in deep neural networks through artificial mental rotation, *arXiv preprint arXiv:2311.08525* .
- Tzeng, E., Hoffman, J., Saenko, K. and Darrell, T. [2017]. Adversarial discriminative domain adaptation, *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7167–7176.
- van der Wel, E. and Ullrich, K. [2017]. Optical music recognition with convolutional sequence to-sequence models, *arXiv preprint arXiv:1707.04877* .
- Vanschoren, J., van Rijn, J. N., Bischl, B. and Torgo, L. [2013]. Openml: Networked science in machine learning, *SIGKDD Explorations* **15**(2): 49–60.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. and Polosukhin, I. [2017]. Attention is all you need, *Advances in Neural Information Processing Systems* **30**, pp. 5998–6008.
- Viaene, S., Dedene, G. and Derrig, R. A. [2005]. Auto claim fraud detection using bayesian learning neural networks, *Expert Systems with Applications* **29**(3): 653–666.
- Waibel, A. [1987]. Phoneme recognition using time-delay neural networks, *Neural Computation* .
- Wang, D., Khosla, A., Gargeya, R., Irshad, H. and Beck, A. H. [2016]. Deep learning for identifying metastatic breast cancer, *arXiv preprint arXiv:1606.05718* .
- West, M. L. [1994]. The babylonian musical notation and the hurrian melodic texts, *Music & letters* **75**(2): 161–179.

- Worrall, D. E., Garbin, S. J., Turmukhambetov, D. and Brostow, G. J. [2017]. Harmonic networks: Deep translation and rotation equivariance, *IEEE conference on computer vision and pattern recognition*, pp. 5028–5037.
- Wu, K., Otoo, E. and Suzuki, K. [2009]. Optimizing two-pass connected-component labeling algorithms, *Pattern Anal. Appl.* .
- Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K. et al. [2016]. Google’s neural machine translation system: Bridging the gap between human and machine translation, *arXiv preprint arXiv:1609.08144* .
- Xia, G., Bai, X., Ding, J., Zhu, Z., Belongie, S. J., Luo, J., Datcu, M., Pelillo, M. and Zhang, L. [2017]. DOTA: A large-scale dataset for object detection in aerial images, *arXiv preprint arXiv:1711.10398* .
- Xiao, J., Hays, J., Ehinger, K. A., Oliva, A. and Torralba, A. [2010]. Sun database: Large-scale scene recognition from abbey to zoo., *Proceedings of the IEEE conference on computer vision and pattern recognition* pp. 3485–3492.
- Xie, S., Girshick, R., Dollár, P., Tu, Z. and He, K. [2017]. Aggregated residual transformations for deep neural networks, *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1492–1500.
- Yetgin, Ö. E., Gerek, Ö. N. and Nezh, Ö. [2017]. Ground truth of powerline dataset (infrared-ir and visible light-vl), *Mendeley Data* **8**.
- Yosinski, J., Clune, J., Bengio, Y. and Lipson, H. [2014]. How transferable are features in deep neural networks?, *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pp. 3320–3328.
- Zamir, A. R., Sax, A., Shen, W. B., Guibas, L. J., Malik, J. and Savarese, S. [2019]. Taskonomy: Disentangling task transfer learning, in S. Kraus (ed.), *International Joint Conference on Artificial Intelligence 2019*, pp. 6241–6245.
- Zeiler, M. D. and Fergus, R. [2014]. Visualizing and understanding convolutional networks, in D. J. Fleet, T. Pajdla, B. Schiele and T. Tuytelaars (eds), *13th European Conference on Computer Vision, Proceedings, Part I*, pp. 818–833.
- Zela, A., Klein, A., Falkner, S. and Hutter, F. [2018]. Towards automated deep learning: Efficient joint neural architecture and hyperparameter search, *arXiv preprint arXiv:1807.06906* .

- Zhang, C., Bengio, S., Hardt, M., Recht, B. and Vinyals, O. [2017]. Understanding deep learning requires rethinking generalization, *5th International Conference on Learning Representations*.
- Zhang, W. [1988]. Shift-invariant pattern recognition neural network and its optical architecture, *Annual Conference of the Japan Society of Applied Physics*.
- Zoph, B. and Le, Q. V. [2017]. Neural architecture search with reinforcement learning, *Proceedings of International Conference on Learning Representations (ICLR)*.
- Zoph, B., Vasudevan, V., Shlens, J. and Le, Q. V. [2018]. Learning transferable architectures for scalable image recognition, *2018 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8697–8710.

Figures

3.1	A typical image and ground truth from the <i>DeepScores</i> dataset (left), next to examples from the MS-COCO (3 images, top right) and PASCAL VOC (2 images, bottom right) datasets. Even though the music page is rendered at a much higher resolution, the objects are still smaller; the size ratio between the images is realistic despite all images being downsampled.	15
3.2	Examples for each of the 118 classes present in <i>DeepScores</i> , ordered by their frequency of occurrence. Even though the resolution is reduced for this plot, some of the larger symbols like brace (row 2, column 6) or gClef (row 1, column 7) are only shown partially to keep a fixed size for each patch. The symbols' full resolution in the dataset is such that the inter-line distance between two staff lines amounts to 20 pixels.	21
3.3	Histogram for the distribution of symbols over all images (logarithmic scale on abscissa, ordinate weighted to give unit area). The majority of images contain from 100 to 1000 objects.	22
3.4	Examples for the different flavors of ground truth available in <i>DeepScores</i>	23
3.5	The same patch, rendered using five different fonts.	24
3.6	The possible size difference of objects in music notation, illustrated by brace and augmentationDot.	25
3.7	Examples for the importance of context for classifying musical symbols: in both rows, the class of otherwise similar looking objects changes depending on the surrounding objects.	25
4.1	Illustration of the DWD network and its sub-components together with input and outputs. The outputs have been cropped to improve visibility	30

4.2	Illustration of the watershed transform applied to different one-dimensional functions.	33
4.3	Detection results for <i>DeepScores</i> and <i>MUSCIMA++</i> examples, drawn on crops from corresponding input images.	40
4.4	Estimate of a class map \hat{M}^c for every input pixel with the corresponding <i>MUSCIMA++</i> input overlaid.	41
4.5	Typical input snippet used by Pacha et al. [Pacha, Choi, Coüasnon, Ricquebourg and Zanibbi, 2018]	43
4.6	Evolution of $loss^b$ (on the ordinate) of a sufficiently trained network, when training for another 8000 iterations (on the abscissa).	43
5.1	Two examples of slanted symbols with their corresponding orthogonal bounding boxes (blue) and oriented bounding boxes (yellow). Orthogonal bounding boxes contain a significant amount of background pixels (left) and can have ample overlap with other bounding boxes (right). Oriented bounding boxes reduce these issues.	48
5.2	Slurs (yellow) and ties (blue) can vary significantly in size, ranging from relatively short instances (top) to almost as wide as the entire staff (bottom). Depicted music are excerpts of “You make it real” by James Morrison.	50
5.3	An example of the instance segmentation ground truth. Every symbol occurrence has its own color due to the encoding of instance information as color values. Color differences have been exaggerated for better readability.	53
5.4	Example detection results of the two provided baseline models from the <i>DeepScoresV2</i> dataset. Both are excerpts from full page detections, cropped for readability.	55
5.5	Example snippets from two <i>RealScores</i> pages with ground truth annotations overlaid.	57
5.6	Example blank pages.	58
5.7	<i>ScoreAug</i> examples (top right, bottom row) derived from the same synthetic sample (top left).	59
6.1	Schematic overview of the <i>Portfolio Hyperband</i> workflow.	70

6.2	Vision: Learning curves as function of training time for MobileNetV2 with varying layout of the fully connected classifier layer(s): 1 layer with 64 neurons (1×64 , red), 2×64 (blue), 1×1024 (yellow) and 2×1024 (green), shown for image datasets Chucky (left) and Pedro (right).	75
6.3	Sponge effect in vision and audio: Evidence for the increased sample efficiency of larger fully connected classifiers (“sponge effect”), shown by the averaged performance of eight different classifier designs on Vision and Audio data.	76
7.1	Is a CNN <i>architecture</i> that performs well on ImageNet automatically a good choice for a different vision dataset? This plot suggests otherwise: It displays the relative test errors of 500 randomly sampled CNN architectures on three datasets (ImageNet, Powerline, and Insects) plotted against the test error of the same architectures on ImageNet. The architectures have been trained from scratch on all three datasets. Architectures with low errors on ImageNet also perform well on Insects, on Powerline the opposite is the case.	81
7.2		87
7.3		87
7.4	(A) Example images from each dataset. Images of Cifar10/100 are magnified fourfold, the rest are shown in their original resolution (best viewed by zooming into the digital document). (B) The structure of models in the AnyNetX design space, with a fixed stem and a head, consisting of one fully-connected layer of size c , (where c is the number of classes). Each stage i of the body is parametrised by d_i, w_i, b_i, g_i , the strides of the stages are fixed with $s_1 = 1$ and $s_i = 2$ for the remainder.	87
7.5	Error of all 500 sampled architectures on subsampled (by number of classes) versions of ImageNet (y-axis) plotted against the error of the same architectures on regular ImageNet (x-axis). The top 10 performances on the target dataset are plotted in orange and the worst 10 performances in red.	90
7.6	Test errors of all 500 sampled architectures on target datasets (y-axis) plotted against the test errors of the same architectures on the ImageNet-X (x-axis). The top 10 performances on the target dataset are orange, the worst 10 performances red.	91

7.7	The errors of all 500 architectures on Cifar10, Natural, and Powerline plotted against the errors on ImageNet (top row), ImageNet-1000-10 (middle row) and ImageNet-10 (bottom row). We observe that class-wise downsampling has the largest positive effect on error correlation.	92
7.8	Test errors of all 500 sampled architectures on target datasets (y-axis) plotted against the test errors of the same architectures (trained and tested) on ImageNet (x-axis). The top 10 performances on the target datasets are plotted in orange and the worst 10 performances in red.	94
7.9	depths	95
7.10	widths	95
7.11	Errors of all 500 sampled architectures on ImageNet, Insects, HAM10000, Powerline, Natural, and Cifar100 (x-axis) plotted against the cumulative block (A) depths and (B) depths (y-axis).	95
8.1	An example image from ImageNet (left) and a version of the same image with its corners masked, to allow for non-obvious and reversible rotations (middle). The top-1 classification error on ImageNet aggregated across 8 CNN architectures for upright (orange) and rotated (blue) training, revealing a training slow-down for rotated inputs (right).	99
8.2	Architecture of our artificial mental rotation module for CNNs. The base CNN, in this case a ResNet, is shown in grey. The components of the AMR module are shown in blue. The information flow in stage 1 (angle classification) is purple while the information flow in stage 3 (image classification) is shown in orange.	103
8.3	Polar plots of ImageNet top-1 accuracies by angle, averaged across architectures (except ViT). The performances of the upright base models are shown in solid orange, the rotated training base models are shown in dash-dotted green, and AMR performance (averaged across both epoch regimes) is shown in dotted blue lines.	107
8.4	Polar plots of Stanford Cars (top row) and Oxford Pet (bottom row) top-1 accuracies analogous to the ones shown above for ImageNet (see Figure 8.3).	108

8.5	Photographs of two printed ImageNet validation samples taken at 12 different angles. Both samples are shown in their original state (raw) and after the mental rotation step (corrected). The color of the masked-out region indicates if the corresponding image has been correctly classified. The mental rotation and classification steps have been performed by ResNet50 + AMR33.	111
8.6	Two rotated examples (rows) from the CoCo validation set with their corresponding ground truth, segmentation output, and AMR-corrected segmentation output (columns). The segmentation on the rotated image exhibits bad performance in both samples. In row one, the persons are segmented fairly well but the table is missing fully, in row two the segmentation fails almost completely. The column AMR-segmentation shows the output for the images that have been un-rotated using AMR. The AMR module identified the correct rotation angle in both cases, which lead to significantly improved segmentation performance (again in both cases). . . .	112
A.1	The Cifar10 test errors of all 500 architectures plotted against ImageNet (top row) and ImageNet-10 (bottom row), shown for our original Cifar10 training (left column), training with a Cifar10 specific stem in the architecture (middle column), and training for 200 epochs, which is roughly 6 times longer (right column). The plots show that the error correlation with ImageNet-10 is much larger in all three cases, confirming that optimizing for individual Cifar10 performance does not alter our core result.	120
A.2	Cifar10 test error curves of 20 randomly sampled architectures trained over 200 epochs (left). The same error curves but cut to epochs 30 to 200.	121
A.3	Test error curves of the five best and five worst models on Powerline and Natural, respectively, when training is continued to epoch 100	122
A.4	Error distributions on Cifar10 of two architectures (122, 147) both trained from scratch 250 times as well as the Cifar10 error distribution of all 500 architectures. The plot shows that the variability caused by changing architecture is much larger than the one caused by random training effects.	123
A.5	Top-1 error plotted against top-5 error of all 500 architectures on ImageNet, Cifar100, and Insects. The plots reveal that on all three datasets the errors have a very close relationship: it is not perfectly linear but is monotonically ascending	123

A.6	Training errors of the sampled architectures (x-axis) plotted against the cumulated block <i>depth</i> for the 3 datasets that have the lowest test errors on shallow architectures. We observe that for all three datasets shallow architectures also have the lowest training errors. Therefore overfitting is not the cause of this behaviour.	124
A.7	Class distributions of MLC2008, HAM10000, and their balanced versions.	125
A.8	Errors of all 500 sampled architectures on MLC2008-balanced and HAM1000-balanced (y-axis) plotted against the errors of their unbalanced counterparts (x-axis). The top 10 performances on the target dataset are plotted in orange, the worst 10 performances in red. We observe a clear positive correlation for both datasets, hence we conclude that the dataset imbalance has a limited impact on the APRs.	125
A.9	Errors form all 500 architectures trained from scratch (blue) as well as the same architectures pretrained on ImageNet (green), plotted against the respective ImageNet errors. We observe that the error correlation with ImageNet increases relative to the performance gain due to pretraining.	126
A.10	Configurations of the top-performing architectures, with the four stages depicted on the x-axis and the parameter values on the y-axis. The best architectures are shown in blue, the mean of the top 15 architectures is depicted in orange with with a vertical indication of one standard deviation.	127
A.11	Matrix of error scatterplots of all datasets except Concrete (The first row replicates plots shown in Figure 7.8).	128
A.12	Polar plots of Stanford Cars top-1 AMR accuracies, shown individually by training duration (dashed vs dotted) and base model architecture (color coded) instead of averaged.	129
A.13	Top-1 angle accuracies for ResNet-18+AMR with the information flow from the base network restricted. Legend indicates which channels (see Figure 2 in the main manuscript) from the base network are open.	131
A.14	Example printout featuring helper lines to facilitate photographs at exact angles.	132
A.15	All re-digitized photos raw (top rows) and after AMR (bottom rows) with their corresponding classifications color coded.	133

Tables

3.1	Information about the number of classes, images and objects for some of the most common used datasets in computer vision. The number of pixels is estimated due to most datasets not having fixed image sizes. We used the SUN 2012 object detection specifications for SUN, and the statistics of ILSVRC 2014 [Russakovsky et al., 2015] detection task for ImageNet.	19
3.2	Statistical measures for the occurrences of symbols per musical sheet and per class (rounded).	22
4.1	AP with overlap 0.5 and overlap 0.25 for the twenty best detected classes of the <i>DeepScores</i> dataset.	38
4.2	AP with overlap 0.5 and overlap 0.25 for the twenty best detected classes from <i>MUSCIMA++</i> .	39
5.1	Average background area reduction for selected classes and average overall reduction. “Background pixel ratio” shows what percentage of pixels within a bounding box is part of the background rather than the foreground.	49
5.2	Summary of changes to symbol classes in DeepScoresV2. Most notable is the addition of hairpins, beams, slurs, and ties. Additionally, some names have been changed to become more consistent, and certain compound symbols have been split into their component symbols for added robustness. Classes that do not occur in the dataset have been removed.	51
5.3	Classwise Average Precision at 0.5 overlap (AP0.5) as well as Mean Average Precision (mAP) of DWD and Faster R-CNN.	56

5.4	Probabilities of augmentations as part of <i>ScoreAug</i> that can be applied to either the blanks, synthetic scores, or both at the same time. Note that P_{aug} decides how likely any other augmentations (after the salt and pepper noise) will be applied, in order to not only feed <i>ScoreAugmented</i> samples to the model. Our final model uses $P_{\text{snr}} = 0\%$, $P_{\text{aug}} = 30\%$, $P_{\text{blur}} = 10\%$.	59
5.5	The AP for the baseline model and models with <i>ScoreAug</i> and <i>Finalise</i> data augmentation on the <i>DeepScoresV2</i> and the <i>RealScores</i> datasets.	60
6.1	Performance of selected automated machine learning algorithms on AutoML challenge data sets [Guyon et al., 2017]. "Test" refers to our own test split; "Time" is in minutes; "Portfolio Hyperband" only tested on binary classification.	71
6.2	Vision: Summary of vision datasets.	73
6.3	Vision: Numerical evaluation of different network architectures on the image and video training datasets.	74
6.4	Vision: Comparison of different pooling methods for image and video classification, using the MobileNetv2 architecture.	75
7.1	Meta data of the used datasets.	85
7.2	Dataset-specific experimental settings.	86
7.3	Comparison of error correlations between target datasets and ImageNet as well as the closest ImageNet-X member.	93
7.4	Correlation of observed error rates with the cumulative block depth and width parameters for all ImageNet-X datasets.	96
8.1	ImageNet top-1 accuracies. Upright testing (up) of the upright trained base model is assumed to be the performance ceiling (% ceil). Average (by angle) rotated accuracies (rot) are given for the upright and rotated trained base models as for AMR 33 epochs and AMR 5 epochs.	105
8.2	Stanford Cars and Oxford Pet top-1 accuracies averaged across all architectures, columns are analogous to the Table 8.1 shown above for ImageNet.	106
8.3	Top-1 accuracies of ResNets on upright (up) and rotated (rot) ImageNet, accompanied with breakpoints (BP) that signify the share of rotated data in the test set necessary for alternative methods (rotated training, AMR) to outperform upright training.	109

8.4	Top-1 accuracies on rotated MNIST for ResNet-18 based methods as well as related works, accompanied by ResNet-18 upright top-1 accuracy as a baseline.	109
A.1	Top-1 error of reference network implementations [Radosavovic et al., 2020] for Cifar10.	120
A.2	Stanford Cars (top) and Oxford Pet (bottom) top-1 accuracies for all architectures analogous to the one shown above for ImageNet (see Table 8.1).	130
B.1	Links to wandb.ai projects containing hyperparameters and relevant metrics of all training and evaluation runs created for this work.	137