Universiteit Leiden
The Netherlands

Learning from small samples

Kocaman, V.

Citation

Kocaman, V. (2024, February 20). *Learning from small samples*. Retrieved from https://hdl.handle.net/1887/3719613

Version:	Publisher's Version
License:	Licence agreement concerning inclusion of doctoral thesis in the Institutional Repository of the University of Leiden
Downloaded from:	https://hdl.handle.net/1887/3719613

Note: To cite this publication please use the final published version (if applicable).

Learning From Small Samples

Proefschrift

ter verkrijging van

de graad van doctor aan de Universiteit Leiden,

op gezag van rector magnificus prof.dr.ir. H. Bijl,

volgens besluit van het college voor promoties

te verdedigen op dinsdag 20 februari 2024

klokke 13:45 uur

door

Veysel Kocaman

geboren te Adiyaman, Turkije

in 1981

Promotor:
Prof.dr. T.H.W. Bäck

Co-promotor:
Dr. O.M. Shir (Tel-Hai College, Israel)

Promotiecommissie:
Prof.dr. A. Plaat
Prof.dr. J. Batenburg
Prof.dr. E. Hart (Edinburgh Napier University, UK)
Prof.dr. P. Kerschke (Technical University Dresden, Germany)
Dr. D.M. Pelt
Dr. N. van Stein

Copyright © *2024 Veysel Kocaman* All Rights Reserved.

This dissertation was made possible through the support of Leiden Institute of Advanced Computer Science (LIACS), Leiden University, which provided me with a part-time position, thereby facilitating my research and studies as an external PhD candidate. This arrangement was essential in enabling the completion of my doctoral studies without external grant funding.

No good can ever come from deviating from the path that you were destined to follow.
–Robert Greene, Mastery

With a heavy heart and profound respect, I dedicate this dissertation to my dear father, whose passing just a few months before the completion of my PhD journey has deeply affected me. The memory of his warm and reassuring smile is ever-present in my thoughts, providing comfort and inspiration in moments of reflection. His absence in the final stages of this academic pursuit is a profound sorrow. For six years, distance separated us, and I had always hoped he would be present to witness this milestone. His enduring love and the wisdom he shared continue to guide and inspire me. His smile, a symbol of joy and love, remains a vivid and cherished memory. As I reach this significant milestone, I do so with the image of his smiling face in my heart, hoping he is watching over me, feeling proud and fulfilled. His spirit has been a guiding light in the most challenging phases of this journey, making this accomplishment not just mine, but ours.

Contents

1 Introduction **1**

1.1 Background 1

1.2 Objectives 2

1.2.1 Research Questions 4

1.3 Outline of the Dissertation 4

1.4 Author's Contributions 5

1.5 Other Work by the Author 6

2 Preliminaries **9**

2.1 Definition of Terms 9

3 Dealing with Small Data in Machine Learning **15**

3.1 Introduction 16

3.2 Handling Small Data 19

3.2.1 Overfitting and Generalization 21

3.3 Approaches to tackle small data problems 26

3.3.1 Data selection and preprocessing 27

3.3.2 Incorporating domain, prior and context knowledge 28

3.3.3 Picking the right approach 31

3.3.4 Ensemble methods 32

3.3.5 Transfer learning 33

3.3.6 Parameter initialization 39

3.3.7 Loss function reformulation 40

3.3.8 Regularization techniques 41

3.3.9 Data augmentation 43

3.3.10 Synthetic data generation 46

3.3.11 Problem reduction 48

3.3.12 Optimization techniques 50

3.3.13 Using physics-informed neural networks 54

3.3.14 Unsupervised learning techniques 56

3.3.15 Semi-supervised learning 56

3.3.16 Self-supervised learning 60

3.3.17 Zero-shot, one-shot and few-shot learning 61

3.3.18 Meta learning 63

3.3.19 Harnessing model uncertainty 64

3.3.20 Active learning 66

3.3.21 Self-learning 71

3.3.22 Multi-task learning 71

3.3.23 Symbolic learning 73

3.3.24 Hierarchical learning 75

3.3.25 Knowledge distillation based learning 75

3.4 Dealing with imbalanced data 76

3.5 Anomaly Detection as a Small Data Problem 78

3.6 Conclusion 80

4 The Role of Final Batch Normalization Layer **83**

4.1 Background 84

4.2 Related Work and Prior Art 87

4.3 Implementation Details and Experimental Results 89

4.3.1 Dataset Details 89

4.3.2 Adding a Final Batch Norm Layer Before the Output Layer 91

4.3.3 Experimentation Subject to Different Configurations 95

4.4 Discussion 101

4.4.1 Additional Experiments 106

4.5 Conclusions 106

5 The Role of Self-Supervised Learning **111**

5.1 Self-Supervised Learning 111

5.1.1 Motivation 111

5.1.2 Background 113

5.1.3 Contrastive Self-Supervised Learning 116

5.2 Preliminary: SSL applied to small subsets of Plant Village data . . 130

5.3 Salient Image Segmentation as a Surprising Player in SSL 132

5.3.1 Introduction: Salient Image Segmentation 132

5.3.2 Background 137

5.3.3 Related Work 142

5.3.4 Implementation, Setup & Results 144

5.3.5 Discussion 151

5.3.6 Conclusions 153

6 Conclusions **163**

7 Bibliography **169**

English Summary **197**

Nederlandse Samenvatting **199**

Acknowledgements **201**

About the Author **203**

Chapter 1

Introduction

1.1 Background

Machine Learning (ML) and Deep Learning (DL) have proven to be powerful tools for solving complex real-world problems. These techniques have been applied to a wide range of domains, including speech recognition, computer vision, natural language processing, recommendation systems and more. However, the performance of these models relies heavily on the availability of large amounts of annotated data, which is often a key barrier for companies to adopt machine learning. Labeling data gets expensive, and the difficulties of sharing and managing large datasets for model development make it a struggle to get machine learning projects off the ground in practical applications.

Labeled data is a group of samples that have been tagged with one or more labels. After obtaining a labeled dataset, machine learning models can be applied to the data so that new, unlabeled data can be presented to the model and a likely label can be guessed or predicted for that piece of unlabeled data. When the amount of available labeled data is limited, ML models tend to overfit on the training data, leading to poor generalization performance on unseen data whereas human can learn new concepts with just a few example and can often generalize successfully

from just a single example. This is particularly problematic in fields where data collection is time-consuming and expensive, such as medicine and biology. In these domains, the limited availability of labeled data restricts the development of powerful models that can provide meaningful insights and predictions.

In addition to the challenges associated with small datasets, ML models also face the problem of imbalanced data. Imbalanced data refers to a scenario in which the number of observations belonging to one class is significantly lower than those belonging to the other classes. This is a common problem in business contexts where accurate prediction is crucial, such as detecting fraudulent transactions, identifying rare diseases, and predicting customer churn. Standard ML algorithms may not produce accurate results when applied to imbalanced datasets, as they are designed to reduce error rather than consider the class distribution or balance of classes.

In short, imbalanced data concept refers to the situation where the number of examples from each class in a dataset is highly imbalanced. This can lead to biased models that perform poorly on the underrepresented class. Anomaly detection is another important challenge associated with small datasets, where the goal is to identify data instances that deviate from the normal behavior.

In order to overcome these challenges, various approaches have been proposed to effectively learn from small datasets in ML. These include data selection and preprocessing, incorporating domain, prior and context knowledge, ensemble methods, transfer learning, parameter initialization, loss function reformulation, regularization techniques, data augmentation, synthetic data generation, and more. The use of these techniques enables the development of models that are capable of effectively learning from limited amounts of data and generalizing to unseen data.

1.2 Objectives

The main objective of this dissertation is to explore various approaches for effectively learning from small datasets in ML, and to address the challenges of imbalanced data and anomaly detection. This will contribute to the development of more robust and efficient models that can be applied to a wide range of real-world

problems where the availability of labeled data is limited. The specific objectives are as follows:

- To identify and analyze the problems associated with small datasets and how they affect the performance of ML models.
- To review and evaluate different techniques for handling small datasets, including data selection and preprocessing, incorporating prior knowledge, and the use of ensemble methods.
- To examine transfer learning and how it can be used to tackle small data problems.
- To investigate various optimization techniques, such as parameter initialization and loss function reformulation, and how they can be used to improve the performance of ML and DL models on small datasets.
- To study regularization techniques and how they can be used to prevent overfitting in ML and DL models.
- To explore data augmentation and synthetic data generation as potential solutions for small dataset problems.
- To evaluate the performance of self-supervised, semi-supervised, and unsupervised learning techniques on small datasets.
- To investigate the potential of using physics-informed neural networks, meta learning, and active learning to handle small data problems.
- To analyze the problem of imbalanced data and how it can be addressed in small sample settings.
- To examine the challenges of anomaly detection as a small data problem and potential solutions.

The proposed research will provide a comprehensive overview of the challenges and solutions associated with learning from small datasets in ML, and will contribute to the development of more effective and efficient models for this task.

1.2.1 Research Questions

RQ1 (Chapter 3) What are the current methods and techniques used to effectively learn from small datasets in Machine Learning and overcome the challenges posed by small data and extreme imbalance?

RQ2 (Chapters 4) Being one of the most effection regularization technique, how does adding a batch normalization layer just before the softmax output layer in modern CNN architectures affect the training time and test error for minority classes in highly imbalanced datasets, and what is the impact of this technique on the overall performance of the model in terms of recall for the minority class?

RQ3 (Chapters 5) How does the use of salient image segmentation as an augmentation policy in Self-Supervised Learning (SSL) impact the representation and generalization capabilities of images in downstream tasks such as image segmentation, and how does this method compare to other commonly used augmentation policies in SSL?

1.3 Outline of the Dissertation

Chapter 3 provides an overview of the challenges that arise in machine learning when dealing with small data. It starts by discussing the problem of overfitting and generalization, and how they can be mitigated when the data is limited. Then, the chapter delves into the various approaches that have been proposed to effectively learn from small datasets. Handling small data includes a detailed discussion of various techniques that can be used to overcome the limitations posed by small datasets. These techniques are grouped under several headings, such as data selection and preprocessing, incorporating domain, prior, and context knowledge, ensemble methods, transfer learning, parameter initialization, loss function reformulation, regularization techniques, data augmentation, synthetic data generation, problem reduction, optimization techniques, and more.

This chapter also includes a discussion of the various approaches that have been proposed to tackle small data problems, with a focus on how they can be used to effectively learn from limited amounts of data. These approaches include, but

are not limited to, using physics-informed neural networks, unsupervised learning techniques, semi-supervised learning, self-supervised learning, zero-shot, one-shot, and few-shot learning, meta-learning, harnessing model uncertainty, active learning, self-learning, multi-task learning, symbolic learning, hierarchical learning, and knowledge distillation based learning.

Chapter 4 covers the problem of learning from data that is highly skewed towards one class, and the various techniques that have been proposed to mitigate this problem. This chapter mainly focuses on the impact of batch normalisation on learning from small datasets. It provides experimental evidence of the positive impact of adding an additional batch normalisation layer just before the softmax output layer on reducing the training time and test error for minority classes in a highly imbalanced dataset. The results show that this approach can lead to a significant improvement in the performance of the model, particularly when a high recall is desired for the minority class.

Chapter 5 focuses on the impact of self-supervised learning on learning from small datasets. The chapter provides experimental evidence of the positive impact of using salient image segmentation as an augmentation policy in self-supervised learning, when the downstream task is image segmentation. The results indicate that using this augmentation policy leads to better image representations, as compared to using default augmentations or no augmentations at all. The chapter concludes with a discussion of the potential of self-supervised learning in mitigating the limitations posed by small datasets.

1.4 Author's Contributions

The main contributions of the author of this dissertation are the following:

[1] Veysel Kocaman, Ofer M Shir, and Thomas Bäck. Improving model accuracy for imbalanced image classification tasks by adding a final batch normalization layer: An empirical study. In *2020 25th International Conference on Pattern Recognition (ICPR)*, number 10.1109/ICPR48806.2021.9412907, pages 10404–10411. IEEE, 2021.

[2] Veysel Kocaman, Ofer M Shir, and Thomas Bäck. The unreasonable effectiveness of the final batch normalization layer. In *International Symposium on Visual Computing*, volume 13018, pages 81–93. Springer, 2021.

[3] Veysel Kocaman, Ofer M Shir, Thomas Bäck, and Ahmed Nabil Belbachir. Saliency can be all you need in contrastive self-supervised learning. In *International Symposium on Visual Computing*, pages 119–140. Springer, 2022.

1.5 Other Work by the Author

(Out of topic research activities over the course of PhD)

[4] Veysel Kocaman and David Talby. Biomedical named entity recognition at scale. In *International Conference on Pattern Recognition*, pages 635–646. Springer, Cham, 2021.

[5] Veysel Kocaman and David Talby. Improving clinical document understanding on covid-19 research with spark nlp. *AAAI-21 Workshop on Scientific Document Understanding*, (arXiv preprint arXiv:2012.04005), 2020.

[6] Veysel Kocaman and David Talby. Spark nlp: natural language understanding at scale. *Software Impacts*, 8:100058, 2021.

[7] Veysel Kocaman and David Talby. Accurate clinical and biomedical named entity recognition at scale. *Software Impacts*, 13:100373, 2022.

[8] Veysel Kocaman, Bunyamin Polat, Gursev Pirge, and David Talby. Biomedical named entity recognition in eight languages with zero code changes. In *Proceedings of the Iberian Languages Evaluation Forum (IberLEF 2022)*, volume 3202, 2022.

[9] Veysel Kocaman, Youssef Mellah, Hasham Haq, and David Talby. Automated de-identification of arabic medical records. In *Proceedings of ArabicNLP 2023*, pages 33–40, 2023.

[10] Veysel Kocaman, Hasham Ul Haq, and David Talby. Beyond accuracy: Automated de-identification of large real-world clinical text datasets. In *Machine Learning for Health (ML4H) 2023–Findings track*, 2023.

[11] Veysel Kocaman, Hasham Ul Haq, and David Talby. Beyond accuracy: Automated de-identification of large real-world clinical text datasets. In *Proceedings of The Professional Society for Health Economics and Outcomes Research (ISPOR) Europe 2023*, Copenhagen, Denmark, 2023. Poster presentation.

[12] Sutanay Choudhury, Khushbu Agarwal, Colby Ham, Pritam Mukherjee, Siyi Tang, Sindhu Tipirneni, Veysel Kocaman, Suzanne Tamang, Robert Rallo, and Chandan K Reddy. Tracking the evolution of covid-19 via temporal

comorbidity analysis from multi-modal data. In *AMIA*, 2021.

[13] Syed Raza Bashir, Shaina Raza, Veysel Kocaman, and Urooj Qamar. Clinical application of detecting covid-19 risks: A natural language processing approach. *Viruses*, 14(12):2761, 2022.

[14] Khushbu Agarwal, Sutanay Choudhury, Sindhu Tipirneni, Pritam Mukherjee, Colby Ham, Suzanne Tamang, Matthew Baker, Siyi Tang, Veysel Kocaman, Olivier Gevaert, et al. Preparing for the next pandemic via transfer learning from existing diseases with hierarchical multi-modal bert: a study on covid-19 outcome prediction. *Scientific Reports*, 12(1):1–13, 2022.

[15] Hasham Ul Hak, Veysel Kocaman, and David Talby. Deeper clinical document understanding using relation extraction. In *https://arxiv.org/abs/2112.13259*. Scientific Document Understanding workshop at AAAI 2022, 2021.

[16] Hasham Ul Hak, Veysel Kocaman, and David Talby. Mining adverse drug reactions from unstructured mediums at scale. In *W3PHIAI workshop at AAAI-22*. https://arxiv.org/abs/2201.01405, 2022.

[17] Murat Aydogan and Veysel Kocaman. Trsav1: A new benchmark dataset for classifying user reviews on turkish e-commerce websites. *Journal of Information Science*, 1:https–journals, 2022.

[18] Vikas Kumar, Lawrence Rasouliyan, Veysel Kocaman, and David Talby. Using natural language processing to identify adverse drug events of angiotensin converting enzyme inhibitors. International Forum on Quality and Safety in Healthcare EUROPE 2021, 2021.

[19] A. Emre Varol, Veysel Kocaman, Hasham Ul Hak, and David Talby. Understanding covid-19 news coverage using medical nlp. In *5th International Workshop on Narrative Extraction from Texts (Text2Story)*, 2022.

[20] Hasham Ul Haq, Veysel Kocaman, and David Talby. Connecting the dots in clinical document understanding with relation extraction at scale. *Software Impacts*, 12(100294), 2022.

[21] Vikas Kumar, Lawrence Rasouliyan, Veysel Kocaman, and David Talby. Detecting adverse drug events in dermatology through natural language processing of physician notes. In *36th International Conference on Pharmacoepidemiology & Therapeutic Risk Management*, volume 36, pages https–www, 2022.

1.5. Other Work by the Author

[22] Juan Martinez, Veysel Kocaman, Hasham Ul Haq, and David Talby. Zero-shot information extraction for clinical nlp. [under review].

[23] Arshaan Nazir, Thadaka Kalyan Chakravarthy, David Amore Cecchini, Rakshit Khajuria, Prikshit Sharma, Ali Tarik Mirik, David Talby, and Veysel Kocaman. Langtest: A comprehensive evaluation library for custom llm and nlp models. [under review].

[24] Julio Bonis, Veysel Kocaman, and David Talby. Social determinants of health in clinical narratives: A comprehensive review of pubmed clinical case reports from 1975 to 2022. Available at SSRN: https://ssrn.com/abstract=4590921 or http://dx.doi.org/10.2139/ssrn.4590921, 2022.

Chapter 2

Preliminaries

2.1 Definition of Terms

This chapter serves as a brief descriptions of the terms mentioned within this dissertation. We should note here that this chapter does by no means offer an exhaustive overview of the aforementioned fields, as this would lie out of the scope of the present thesis. Its aim, instead, is to acclimatize the reader to the definitions of terminology that will be recurring in the following chapters.

Machine learning (ML) Machine learning is a subfield of artificial intelligence that focuses on the development of algorithms and statistical models that can learn patterns and relationships in data. These algorithms can then be used to make predictions or decisions about new, unseen data.

Deep learning (DL) Deep learning is a subfield of machine learning that uses neural networks with multiple layers to model and solve complex problems. These deep networks are trained using large amounts of data and can learn hierarchical representations of data, allowing them to perform well on a variety of tasks such as image recognition, natural language processing, and speech recognition.

Overfitting Overfitting occurs when a machine learning model is too complex for the amount of data it is trained on. As a result, the model fits the training

data too closely and does not generalize well to new, unseen data, leading to poor performance.

Underfitting Underfitting occurs when a machine learning model is too simple and cannot effectively capture the underlying patterns in the data. As a result, the model will have poor performance on both the training and test data.

Self-supervised learning (SSL) Self-supervised learning is a form of unsupervised learning where the model learns from input data without explicit supervision, but with the goal of reconstructing the input data in some way. This can be done, for example, by predicting missing parts of an input or by rearranging the input data in a specific order.

Semi-supervised (SeSL) Semi-supervised learning is a form of machine learning that combines both supervised and unsupervised learning. In this setting, a small portion of the data is labeled, and the rest is left unlabeled which is usually large. The model then uses the labeled data for supervised learning and the unlabeled data for unsupervised learning to drive structure from unlabeled data to improve the overall performance of the task.

Unsupervised learning Unsupervised learning is a form of machine learning where the model learns from input data without explicit supervision. The goal is to find patterns or relationships in the data without being told what the target outputs should be. It relies on the model learning the structure of the input data based on the features present in the data via feature extraction, which involves identifying and extracting relevant characteristics from the raw data.

Softmax The softmax function is a mathematical function commonly used in machine learning as an activation function for multi-class classification problems. The function maps input values to a probability distribution over multiple classes, allowing the model to make predictions about the class with the highest likelihood.

Probably Approximately Correct (PAC) The Probably Approximately Correct (PAC) framework is a theoretical framework for understanding the generalization ability of machine learning algorithms. The framework provides a mathematical definition of when a learning algorithm is considered to have good generalization performance, making it a useful tool for comparing and evaluating different algorithms.

Synthetic data Synthetic data is artificially generated data used for training machine learning models. The data can be generated by a variety of methods, including simulation, extrapolation, and statistical sampling. The use of synthetic data can be useful in cases where obtaining real-world data is difficult or impossible.

Data Augmentation Data augmentation is a technique used to artificially increase the size of a dataset by generating new data from existing data. This can be done, for example, by applying random transformations or perturbations to the data, or by generating new samples from a generative model. The goal of data augmentation is to reduce overfitting by providing the model with more diverse training data.

Weight Decay Weight decay is a regularization technique in machine learning that penalizes large weights in the model to reduce overfitting and improve the generalization performance. It involves decreasing the magnitude of the weights over time, effectively reducing the complexity of the model and preventing it from memorizing the training data.

Dropout Dropout is a regularization technique used in deep learning to prevent overfitting by randomly dropping out neurons during training. This random dropout creates a different network architecture for each training iteration, forcing the network to learn more robust features.

Early Stopping Early stopping is a method in deep learning used to prevent overfitting by monitoring the performance of the model on a validation set, and stopping training once the performance on the validation set starts to degrade. This helps to ensure that the model is not overfitting to the training data.

Lottery Ticket Hypothesis The lottery ticket hypothesis is a concept in deep learning that suggests that sparse, randomly initialized neural networks can be trained to perform as well or better than dense networks. The hypothesis states that a small, well-initialized subnetwork of a larger network can be trained to perform well if it has the right connections and weights.

Ensemble Models Ensemble models are machine learning models that combine the predictions of multiple individual models to improve overall performance. The idea behind ensemble models is that by combining the predictions of several models, the strengths of each model can be leveraged to create a more robust overall system.

Transfer Learning Transfer learning is the process of using a pre-trained model on one task as a starting point for training on a different, but related, task. The idea is that the pre-trained model has already learned relevant features for the new task, allowing for faster and more effective training.

Regularization Regularization is a technique used in machine learning to prevent overfitting by adding a penalty term to the loss function during training. The goal is to encourage the model to learn a simpler and more general representation of the data.

Generalization Performance Generalization performance refers to a machine learning model's ability to make accurate predictions on new, unseen data. It is the ability of the model to generalize from the training data to unseen data, and is an important consideration in the design of machine learning algorithms.

Pruning Pruning is a method used to reduce the size and complexity of a machine learning model. The process involves removing unimportant connections or neurons from the model to make it more efficient and improve its generalization performance.

Bayesian Methods Bayesian methods are a family of machine learning algorithms that use Bayesian statistics to model the underlying relationships between inputs and outputs. Bayesian methods allow for the incorporation of prior knowledge and the estimation of uncertainty in the model.

Frequentist Methods Frequentist methods are a family of machine learning algorithms that use frequentist statistics to model the underlying relationships between inputs and outputs. It makes predictions on the underlying truths of the experiment, using only data from the current experiment (the parameters are fixed but unknown). Frequentist methods assume the observed data is sampled from some distribution. For example, in logistic regression the data is assumed to be sampled from Bernoulli distribution, and in linear regression the data is assumed to be sampled from Gaussian distribution.

Artificial General Intelligence (AGI) Artificial General Intelligence (AGI) is a field of artificial intelligence focused on the development of machine systems that have general cognitive abilities, including the ability to understand or learn any intellectual task that a human being can. It is a form of artificial intelligence that goes beyond narrow AI and aims to develop machines that can perform a wide

range of tasks.

L1 Norm Regularization (Lasso) L1 norm regularization, also known as Lasso, is a type of regularization in machine learning that adds a penalty term to the loss function that is proportional to the absolute value of the coefficients of the model. This regularization technique encourages the model to have sparse solutions, meaning that some of the coefficients will be exactly zero, effectively reducing the number of features the model uses. This can improve the interpretability and stability of the model, but may also reduce its accuracy.

L2 Norm Regularization (Ridge) L2 norm regularization, also known as Ridge, is a type of regularization in machine learning that adds a penalty term to the loss function that is proportional to the square of the magnitude of the coefficients of the model. This regularization technique encourages the model to have small, but non-zero coefficients, and helps to prevent overfitting by reducing the magnitude of the coefficients. The regularization term can be controlled by a hyperparameter, which determines the strength of the penalty, and the optimal value of this hyperparameter must be determined through cross-validation.

Chapter 3

Dealing with Small Data in Machine Learning

In this chapter, we embark on a critical exploration of the myriad approaches to address the unique challenges posed by small sample sizes in machine learning. The genesis of this chapter lies in my quest to find more effective means of training predictive models capable of learning efficiently from limited data. My journey began with an in-depth exploration of batch normalization and gradually evolved into an extensive study of Self-Supervised Learning (SSL), culminating in several papers ([1], [2], [3]) on these subjects.

Realizing a gap that there wasn't a single, consolidated source summarizing the myriad methods and historical perspectives on learning from small samples led to the creation of this chapter; a chapter that started as a necessity for my work but grew into something much larger—a detailed survey and review chapter. This chapter, therefore, is more than just a narrative thread in my dissertation; it is a compilation of a journey, an exploration, and a guide, setting the stage for the deeper and more focused explorations while laying the foundational groundwork, essential for understanding the advanced techniques and specific applications discussed in subsequent chapters.

In the following sections, we navigate through the complexities of overfitting and generalization, and how these are magnified in small data contexts. The chapter then unravels a comprehensive suite of approaches, encompassing techniques like ensemble methods, transfer learning, parameter initialization, loss function reformulation, and regularization techniques. We also discuss innovative methods such as data augmentation, synthetic data generation, and various optimization techniques, all tailored to enhance learning from small datasets. The exploration extends to cutting-edge concepts like physics-informed neural networks, unsupervised, semi-supervised, and self-supervised learning, along with zero-shot, one-shot, and few-shot learning, metalearning, and the harnessing of model uncertainty. Furthermore, we delve into the realms of active learning, self-learning, multi-task learning, symbolic learning, hierarchical learning, and knowledge distillation-based learning, each offering unique insights and solutions to the small data challenge.

Additionally, this chapter addresses specific challenges such as dealing with imbalanced data and anomaly detection as a small data problem, further illustrating the multifaceted nature of these challenges in machine learning. These methods serve as the building blocks for the sophisticated approaches examined in Chapters 4 and 5, specifically focusing on imbalanced data and the impact of self-supervised learning. By the end of this chapter, readers will gain a comprehensive understanding of the fundamental principles and techniques that are pivotal in harnessing the power of small datasets for machine learning.

In sum, this chapter is not an isolated exploration but a vital piece of the puzzle, intricately linked to the advanced techniques and case studies elaborated in Chapters 4 and 5. Therefore, it serves as a narrative thread that weaves through the entire dissertation, setting the stage for a deeper dive into specific applications and methodologies in machine learning when confronted with limited data.

3.1 Introduction

Recent advances in implementing deep learning (DL) techniques have demonstrated that modern algorithms and complex architectures can enable machines to perform tasks with a level of proficiency similar to that of humans. However, it is also evident that a large amount of training data plays a crucial role in the success of

DL models.

It is a well known argument, both theoretically and empirically, that for a given problem, with a large enough dataset, different algorithms tend to perform similarly. It is important to note that this large dataset should contain meaningful information rather than noise, in order for the model to effectively learn from it. The access to large datasets has contributed to the dominance of companies such as Google, Meta, Apple and many more in AI research and product development. Although traditional machine learning (ML) typically requires less data than deep learning, the performance of both traditional ML and DL models is similarly impacted by the availability of large datasets. Figure 3.1 illustrates the performance gap of traditional ML to DL models, with the increasingly growing amounts of data, and proposes a qualitative estimation of the gap between large to small companies. Figure 3.1 basically says that with limited data, traditional algorithms exhibit marginally superior performance over deep learning. As the volume of data increases, deep learning outperforms traditional methods significantly.

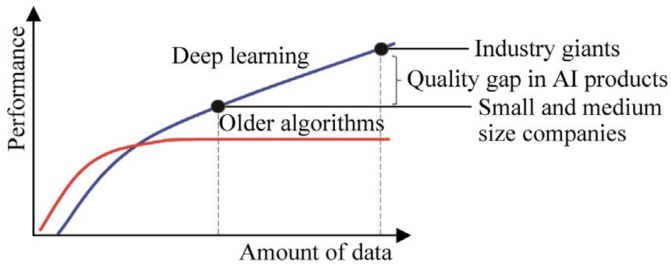


Figure 3.1: A qualitative estimation of performance comparison between traditional ML and deep learning, as well as the performance gap between large to small companies [25]. With limited data, traditional algorithms exhibit marginally superior performance over DL. As the volume of data increases, DL outperforms traditional methods significantly. The red line in this chart indicates the performance of traditional ML algorithms as the amount of data increases. It shows that traditional algorithms start off with a higher performance than DL with small data sizes, but their performance improvement plateaus as the amount of data grows, unlike DL which continues to improve.

Unlike traditional machine learning, human beings are capable of learning about new objects with just a few examples using various approaches, such as relying

on prior experience, comparing known and unlabeled examples, and internalizing semantic descriptions of the object in question. The human learning process, even at early stages, is of particular interest for machine learning, as a ML model initially lacks any prior experience [26]. If we can develop ML models that learn in a similar way, we may be able to approximate human-like learning with small data. Many ML techniques for handling small data are inspired by the human approach, such as data-driven approaches that use prior experience for low-shot problems or learn how to learn tasks, or approaches that replicate human recognition of objects with a few examples through human labeling or semantic description.

The distinction between small and large datasets can be examined further within the context of the Probably Approximately Correct (PAC) model [27]. The PAC model aids in determining the probable characteristics that an algorithm can learn, taking into consideration factors such as sample complexity, time complexity, and space complexity for the algorithm.

PAC basically links the sample size to error rates and to the general success probability, and says that given data X, we expect our algorithm A to output a correct (up to error ϵ) classification hypothesis with probability of at least $1 - \delta$ (δ : failure probability, well-distributed over X). The sample complexity then becomes a function of ϵ and $\delta : m(\epsilon, \delta) = S$ (required examples). This means that for arbitrarily high probability and low error (arbitrarily small δ and ϵ), we can always find a learning algorithm A and a sample size S that achieves that high probability and low error.

The distinction between small and large datasets in the PAC framework is thus related to the model's success probability and error rates in generalizing from the training data to new, unseen data. The distinction between small and large datasets is a critical factor to consider in the design of PAC models, as it affects the generalization performance of the model and the amount of data required to train it. In the PAC framework, a model is considered to have "learned" a concept if it can correctly (up to an acceptable error rate) classify unseen examples with high probability, under certain assumptions about the distribution of the data. Importantly, the PAC framework provides explicit bounds for the sample sizing as a function of the learner's success probability as well as the expected error rate.

Altogether, if the problem is learnable, then satisfying the PAC's bounds on the sample sizing would guarantee the learner's convergence (an event with a certain probability) to a hypothesis that (approximately) classifies the data.

In other words, if the concept is simple and there is a large amount of data available, then a smaller sample size may be sufficient for the model to generalize accurately. However, if the concept is complex and there is a small amount of data, a larger sample size may be required for the model to generalize accurately.

3.2 Handling Small Data

One crucial question that is often overlooked is why we need for ML to address the problem at hand. Consider a scenario where we want to predict the distance a ball will travel when thrown with a velocity (v) at an angle (θ). Using the principles of projectile motion from high-school physics, we can use the equations to precisely determine the pathway of the ball. In this case, the equation can be considered the model or representation for the task, with the variables v , θ, and g (the *gravitational* acceleration) serving as important features. In situations in which exist a few features and a clear understanding of their impact on the task, it is possible to develop a precise analytical model and solve it instantly (mostly by hand).

Now consider a different scenario in which we want to predict a stock price of a company we would like to invest in. In this case, it is difficult to grasp the full range of factors that might influence stock prices. Without a true model, we can use historical stock prices and various other features, other stock prices, and market sentiment, to discover the latent relationships using a ML algorithm. This illustrates a situation where it can be challenging for humans to understand the intricate relationships between a large number of features, but machines can easily capture them by exploring large amounts of data.

In order to build an effective model, it is generally desirable to minimize both bias and variance. This involves creating a model that not only fits the training data well, but also generalizes well to test or validation data. There are various techniques that can be used to achieve this goal wherein training with a larger

dataset is usually the first one.

Imagine a dataset with a distribution resembling a sinusoidal wave. The chart at the upper left corner of Figure 3.2 shows that there are multiple models that are able to fit the data points well. However, many of these models overfit and do not generalize well to the entire dataset.

In DL, overfitting refers to a situation where a model has learned the training data too well, to the point where it memorizes the data and is not able to generalize to new, unseen data. In other words, the model has a high accuracy on the training data, but poor accuracy on the test data. This occurs because the model is too complex and has too many parameters, so it captures the noise and random variations in the training data.

On the other hand, generalization refers to the ability of a model to accurately make predictions on new, unseen data that is drawn from the same distribution as the training data. It is a key aspect of ML, as the goal is to build models that can generalize to new data, rather than just memorizing the training data.

As the amount of data increases, as shown in the charts in Figure 3.2, the number of models that can fit the data decreases. As the number of data points continues to increase, the true distribution of the data is captured more accurately. This example illustrates how a larger dataset can help a model uncover the true underlying relationships.

Another example of a complex task that is challenging for humans but can be easily tackled by ML algorithms is identifying spam emails. Manually developing rules and heuristics to distinguish spam emails can be time-consuming and difficult to maintain. In contrast, ML algorithms can easily learn these relationships from the data and perform the task more accurately and efficiently. This ability to learn relationships from data rather than relying on explicit rules has contributed to the widespread adoption of ML in a variety of fields and industries.

Deep neural networks have millions of parameters to learn and this means we need a lot of iterations before we find the optimal values. If we have small data, running a large number of iterations can result in overfitting. Large dataset helps us avoid

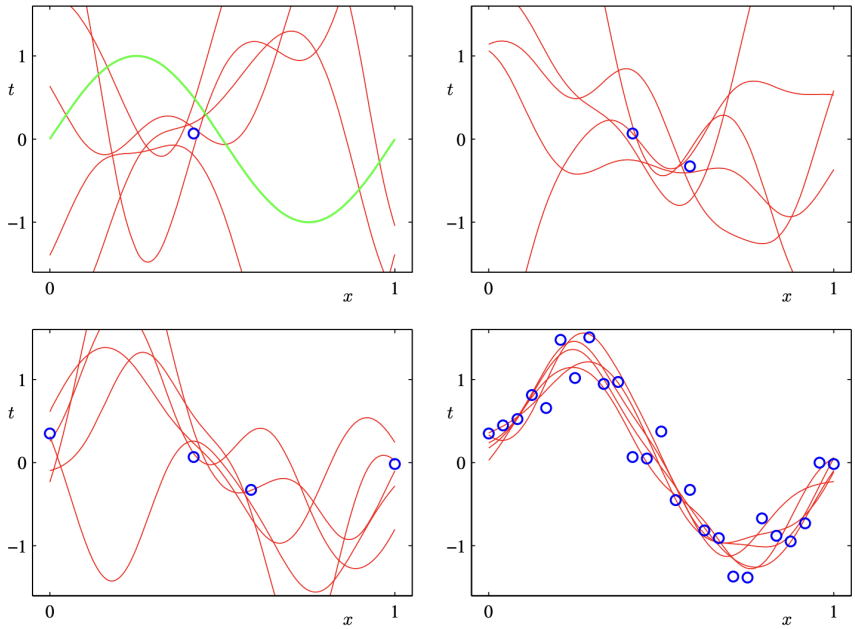


Figure 3.2: Any model can "fit" a single data point (as seen in the upper left corner). As we have more points, less and less models can reasonably explain them altogether. The more data points we have, the better the model capacity to uncover the true underlying relationships; hence less hypotheses we need. As the number of data points continues to increase, the true distribution of the data is captured more accurately (source: [28]).

overfitting and generalizes better as it captures the inherent data distribution more effectively. Due to close relationship of overfitting (caused by the small size of dataset) with the strategies to overcome the lack of enough sample of data points, we will elaborate on the techniques to reduce overfitting (thus enabling generalization) in the following section.

3.2.1 Overfitting and Generalization

The utilization of deep neural networks requires the learning of a vast number of parameters, thus requiring a substantial number of iterations for the optimization course of these parameters. However, in instances where the amount of data that a ML model is trained on is limited, it is more prone to overfitting (especially

when subject to a high number of iterations). This is because there is less feature information to learn from, so the model may end up fitting to the noise or random fluctuations in the training data rather than the underlying pattern. Conversely, utilizing a dataset of ample size can aid in preventing overfitting and yield better generalization by effectively capturing the underlying distribution of the data. As a result, the model may perform well on the training data but poorly on new, unseen data (i.e., when tested/validated).

Reducing overfitting is a crucial step in the training of neural networks as it ensures that the model generalizes well to unseen data. Overfitting can occur when a model is too complex and has seen too little data. To combat this issue, there are several techniques that can be employed from both a dataset perspective and a model perspective.

From a dataset perspective, one effective technique for reducing overfitting is to acquire more labeled data. The more data the model is trained on, the more robust it will be to overfitting. Another technique is to use Data Augmentation and generate synthetic data [29]. This technique allows for the creation of new data points from existing ones, which helps to diversify the training set and reduce overfitting. Additionally, using labeled data from related domains for pretraining via Transfer Learning [30] can be beneficial. By using a model pre-trained on related data, the model will have a better "understanding" of the underlying data distribution, reducing overfitting. Lastly, leveraging unlabeled data for pretraining via Self-Supervised Learning [31] can be a useful technique. By training the model on unlabeled data, the model can learn useful representations of the data, which can be fine-tuned on the labeled data, thus reducing overfitting.

From a model perspective, reducing overfitting can also be achieved by adding regularization to reduce complexity [32]. L2 penalty, weight decay, dropout, and early stopping are all techniques that can be used to reduce model complexity and prevent overfitting [33]. Another technique is to use smaller models, such as the lottery ticket hypothesis [34] and knowledge distillation or to use simpler models that require fewer data points. Lastly, building ensemble models can also be an effective technique for reducing overfitting. Ensemble models [35] combine the predictions of multiple models, which can help to smooth out any overfitting

that may have occurred in individual models (see 3.3.4 for more details regarding ensemble techniques).

In conclusion, reducing overfitting is a crucial step in the training of neural networks, and there are several techniques that can be employed from both a dataset perspective and a model perspective to achieve this goal. We will cover the techniques to reduce overfitting in detail in the following sections.

Recent research [36] has been studying the effect of **pruning** on generalization performance and the findings are intriguing. **Pruning** is a technique to combat overfitting, which is when a model learns the training data too well, including its noise and outliers, to the detriment of its performance on unseen data. By removing some of the weights, pruning simplifies the model, which can help it to generalize better to new data. It has been known for some time that pruning, or producing smaller models, can boost generalization performance. However, other studies have shown that larger, overparameterized models can also improve generalization performance, as seen in double descent and grokking studies. This raises the question of how to reconcile these seemingly contradictory observations.

[36] have found that the reduction of overfitting due to pruning can be partly explained by the improved training process. Pruning involves more extended training periods and a replay of learning rate schedules which can contribute to improved generalization performance. Figure 3.3 illustrates that a pruned model can achieve better generalization performance, not only because of its reduced complexity but also due to the improved training dynamics such as extended training periods and learning rate adjustments. This effect is enhanced in the presence of noisy data, where pruning helps the model to ignore the noise and focus on the underlying patterns, leading to a better generalization on the test data. Additionally, when applied to noisy datasets, the generalization performance improvements due to pruning can be explained by a larger loss on noisy training examples.

Setting larger loss function values on noisy training examples is beneficial because pruned models don't try to fit these noisy examples, which adds a regularizing effect. This is somewhat similar to reducing the width of the layers. This is a new

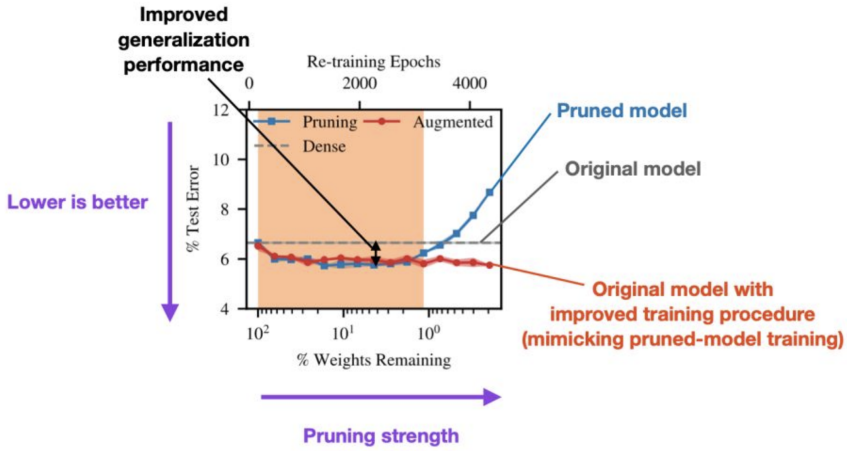


Figure 3.3: Improved training caused by prunning might be responsible for better generalization performance (source: [36]). The blue line represents the pruned model which initially shows improved generalization (lower test error). The retraining and learning rate adjustments during pruning help the model to perform better on test data. The orange line (dense) remains flat across the pruning strength, indicating that the unpruned model's performance does not change with respect to the percentage of weights remaining, as it is not subjected to pruning. The dotted line shows that an improved training procedure can mimic the effects of pruning, leading to a similar generalization performance without actually reducing the model's complexity. This suggests that it's not just the act of pruning that improves performance, but also the associated training techniques. The augmented area suggests that there may be additional methods or data augmentation techniques at play that further improve the generalization performance beyond just pruning.

research area and the findings are very promising, and it will be interesting to see how the research on this subject progresses in the future.

Model averaging is another technique that can be used to mitigate the risk of overfitting and improve generalization performance by combining multiple models. This is achieved by taking the mean or weighted average of the prediction output of each model. While this approach has been shown to be effective in reducing variance and improving generalization, it is important to consider the potential drawbacks, such as the need to maintain and evaluate a large collection of models, which can be computationally expensive and challenging to deploy in a production system.

In terms of generalization, [37] demonstrates that when training neural networks on small synthetically-generated datasets (commonly referred to as algorithmic datasets – a collection of data specifically designed to evaluate the performance of neural networks on tasks that involve symbolic and algorithmic reasoning), unexpected patterns of generalization occur more frequently and drastically compared to datasets obtained from natural sources, even when the performance on the training set is not affected. In [37], as seen in Figure 3.4, the authors show that long after severely overfitting, validation accuracy can sometimes begin to suddenly increase from "chance" level toward perfect generalization, leading to a phenomenon called *grokking* by the authors.

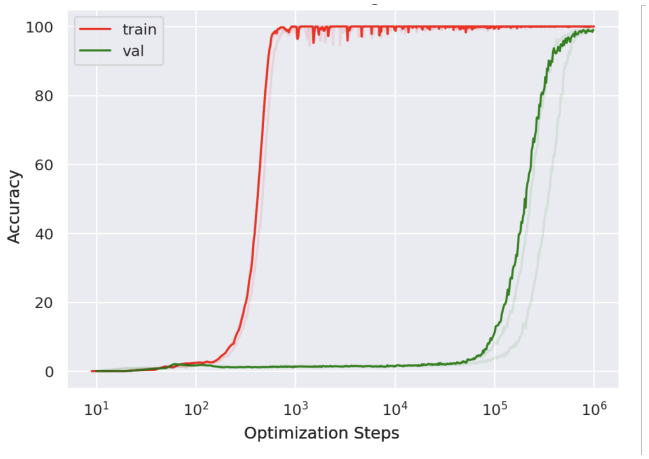


Figure 3.4: A dramatic example of generalization far after overfitting on an algorithmic dataset (a collection of data specifically designed to evaluate the performance of neural networks on tasks that involve symbolic and algorithmic reasoning). The red curves show training accuracy and the green ones show validation accuracy. Training accuracy becomes close to perfect at $< 10^3$ optimization steps, but it takes close to 10^6 steps for validation accuracy to reach that level, and we see very little evidence of any generalization until 10^5 steps (source: [37]).

Another technique to combat overfitting caused by small datasets is regularization, that works by adding a penalty term to the model's cost function that discourages large weights [38]. This helps to constrain the model and make it simpler, which can prevent it from fitting to the noise in the training data (see 3.3.8 for more details on regularization).

We will cover some of these techniques in detail, mostly the ones from a (small) dataset perspective, in the following sections.

3.3 Approaches to tackle small data problems

Small data refers to datasets with a small number of instances, labels, features, or altogether. Dealing with small data poses a challenge in ML as it can lead to overfitting, underfitting, and poor generalization. However, despite the demanding theoretical bounds (e.g., PAC), there are various pragmatic approaches that can be employed to address this challenge and improve the performance of ML models on small data in practice.

Small data can manifest itself in various forms. In some cases, small amount of data that is mostly labelled or unlabelled can be regarded as small data, while in other cases large amount of data that is mostly unlabelled or the number of target classes is partially known (anomaly/ outlier detection) can also be regarded as a small data problem. One of the major challenges of working with small datasets is that the samples may not accurately reflect the underlying distribution of data in the population, leading to difficulties in generalizing the model's performance to new, unseen data.

A small and largely unlabeled dataset is the most infamous version of small data problem wherein both the size of data itself and the labels are not enough to build a reliable model. This situation can pose significant challenges for ML models. The limited amount of data may not provide sufficient information for the model to accurately learn the underlying relationships and patterns in the data. Additionally, the lack of labels can make it difficult for the model to learn from the data, as it lacks the guidance provided by explicit class assignments. This can lead to poor generalization performance on unseen data, as the model may overfit to the limited and potentially noisy examples in the dataset. To mitigate these issues, it may be necessary to carefully select and preprocess the available data, and potentially incorporate additional sources of information, such as domain knowledge.

In summary, there are several approaches that can be employed to deal with small data in machine learning, including regularization, cross-validation, transfer

learning, and data augmentation. Each approach has its own strengths and limitations, and the most appropriate approach will depend on the specific context and goals of the ML task. In the following sections, a few potential approaches that may be useful in this situation will be elaborated.

3.3.1 Data selection and preprocessing

Carefully selecting and preprocessing the available data can help to ensure that the model is able to learn from the most relevant and informative examples. This may involve removing noisy or irrelevant data points, or aggregating or synthesizing additional data to augment the available dataset.

Outliers can have a huge impact on the model and should be identified and removed carefully – overall, it is argued removing the impact of outliers is essential for getting a sensible model with a small dataset. In some cases, outliers could even assist rather than harm the model trained with small datasets [39].

It is hard to avoid overfitting with a small number of observations and a large number of predictors. There are several approaches to feature selection, including analysis of correlation with a target variable, importance analysis, and recursive elimination. It is also worth noting that feature selection will always benefit from domain expertise.

In some cases, the dataset at hand may not exhibit the "small dataset effect", despite its size, if its impact on the learning task is good enough. After all, not all the examples (observations) are equally important and helpful for the model to learn the pattern and generalize over unseen data. Finding the most useful portion of a larger dataset, also known as *coresets*, is another active research area called dataset *prunning*, in which identifying coresets that allow training to approximately the same accuracy as attainable with the original data.

When aiming to achieve a specific target model performance with a limited training data, the question arises: What should be the size of the training subset to achieve a certain performance ? There are a number of different methods (including a prior distribution in Bayesian and frequentist methods that focus on estimation rather than testing) proposed to determine meet this data size requirements [40] [41].

In a similar context, [42] proposed a simple and effective sample size prediction algorithm that conducts weighted fitting of learning curves.

These works attempt to identify examples that provably guarantee a small gap in training error on the full dataset [43]. In this context [44] proposed a scoring method that can be used to identify important and difficult examples early in training, and prune the training dataset with insignificant decline in test accuracy (the examples that can be removed from the training data without hampering accuracy). [44] found out that even at initialization, 50% of the examples can be prunned from the CIFAR-10 dataset without affecting accuracy, while on the more challenging CIFAR-100 dataset, 25% of examples can be prunned resulting in only a 1% drop in accuracy.

3.3.2 Incorporating domain, prior and context knowledge

Human conceptual knowledge has proven difficult for machine systems to replicate in two ways. Firstly, humans are capable of learning new concepts from only a few examples, whereas ML algorithms typically require significantly more examples to perform similarly. This one-shot learning ability allows humans to easily grasp the boundaries of new concepts and make meaningful generalizations, even in the case of children. In contrast, many ML approaches, particularly DL models, require large amounts of data to achieve high levels of performance on tasks such as object and speech recognition. Secondly, human conceptual knowledge tends to be more complex and versatile than that of machines, allowing for the creation of new exemplars, the parsing of objects into parts and relations, and the creation of abstract categories based on existing ones.

One of the key challenges in artificial intelligence and ML is replicating the ability of humans to learn new concepts from just a few examples and to develop abstract, flexible representations. While current ML approaches often require large amounts of data, especially in the case of deep learning, humans are able to learn new concepts and create abstract categories with relatively minimal exposure. Additionally, human learning produces rich representations that can be applied to a wide range of functions, including generating new exemplars and parsing objects into parts and relations, while ML approaches often do not have this capability. A central

question in this field is how to explain these differences in learning between humans and machines, and how to bring the two approaches closer together, particularly given the trade-off between the complexity of a model and the amount of data required for good generalization.

Given this context, Artificial General Intelligence (AGI) may be seen as a potential solution to bridge the gap between human and ML by finding a balance between model complexity and data requirement. AGI [45] refers to a field of AI research that aims to develop algorithms that can perform a wide range of cognitive tasks that are typically performed by humans, including perception, reasoning, learning, and problem solving. The goal of AGI is to create machines that can learn and generalize to new situations, similar to how humans can. The trade-off between the complexity of a model and the amount of data required for good generalization is one of the challenges in achieving AGI. In order to bring the two approaches closer together, researchers are exploring various methods, including developing more sophisticated algorithms, increasing the amount of data available, and improving the interpretability of AI models. AGI has the potential to play a significant role in bridging this gap by enabling algorithms to learn and generalize more like humans.

If there is domain knowledge available that could be used to inform the model's learning process, it may be beneficial to incorporate this information into the model's training. For example, this could involve providing the model with additional constraints or regularization terms to help guide its learning. One way to address the issue of data scarcity is to utilize domain knowledge to restrict the inputs to the model, thereby reducing dimensionality.

Another naive approach would be using rule-based learning, wherein we define rules to arrive conclusions rather than training a model on historical data points. It is also one of the most practical approach to inject weak supervision to the learning process, especially while labelling additional data points (e.g. domain experts write labeling functions that express arbitrary heuristics, which can have unknown accuracies and correlations [46]). Moreover, in order to alleviate the labor-intensive process of manually collecting ground truth labels for ML applications, aggregating multiple sources of weak supervision that is powered by rule based learning reducing the data-labeling bottleneck [47].

Creating rules for labeling data that accurately reflect the characteristics of the data being analyzed usually involves several steps. We start with generating a hypothesis about what the rules should be based on the characteristics of the data (e.g., if the patient's fever is above x and oxygen level is below y, the patient is diagnosed as z). Next, we observe the data to validate the hypothesis and ensure that it accurately reflects the characteristics of the data (using the small sample of observation to validate the rules). Then, we start with simple rules based on the observations made during this process. Finally, we improve the rules over time as more data becomes available and the rules are refined.

The same approach can also be used to reduce false negatives (e.g., choose positive labels when two models disagree) in scenarios such as predicting whether a patient has cancer (it is better to make a mistake telling patients that they have cancer than to fail to detect cancer). Similarly, in order to reduce false positives, choosing negative labels when two models disagree can also be used.

Incorporating domain knowledge through rule-based models have certain limitations. One such limitation is the inability to generalize to unseen data, which can make it difficult to apply these models to new situations. Additionally, it can be challenging to create rules for complex data, and there is no feedback loop to continuously improve the model. Therefore, combining a rule-based model with a ML model may be a more effective approach for data labeling, as it allows for the incorporation of domain expertise while also leveraging the ability of ML models to improve and scale with new data. In order to balance the trade-off between false negatives and false positives, we can decide which type of error to prioritize when combining these two models as briefly mentioned above.

Similarly to context injection, a method of incorporating metadata or contextual information into a ML model in order to improve its accuracy on a specific problem, can also be applied. This approach involves augmenting traditional ML models with contextual information that may be available in the data and can be applied to a wide range of problems, and can be particularly effective when metadata or contextual information is available that is relevant to the task being addressed. The incorporation of contextual information can be achieved through various methods including training a model from scratch with context as an input, using end-to-end

approaches that consider context, or augmenting a pre-trained model by combining context and prediction distributions to make a final prediction.

In another approach called Bayesian program learning (BPL) [48], applying human-level concept learning through probabilistic program induction, the authors construct a model that builds concepts from parts by leveraging prior knowledge in the process. This led to human-level performance and outperforms the DL approaches.

Implicit or explicit sources of domain-knowledge, represented either as logical or numeric constraints (often provided with the help of modification to a loss function) can also be used at the model-construction stage by DNNs as explored in detail by [49]. Implicit Composite Kernel (ICK) [50] is another flexible method that can be used to include prior information or known properties (e.g., seasonality) in neural networks.

3.3.3 Picking the right approach

Deep learning approaches often require a large amount of data to be trained, whereas shallower neural networks and traditional ML methods may require less data and can outperform deep neural networks in scenarios with limited data. In machine learning, small data sets often require models with low complexity or high bias to prevent overfitting. In addition to Support Vector Machines (SVM), decision trees and Gaussian processes (GP), Bayesian methods such as the naive Bayes classifier and ridge regression, are often the best choice for very small data sets, although the results may depend on the choice of prior knowledge. SVM has a particularly good generalization performance in the case of high-dimensional data and a small set of training patterns, without using non-kernel methods [51].

The Naive Bayes algorithm, which is based on the assumption of conditional independence between features given the class variable, is a simple classifier that performs well on small data sets. Linear models and decision trees (or ensembles of them, also known as random forests), which also have a relatively small number of parameters, can also be effective on small data sets. In general, models that have fewer parameters or strong prior assumptions are more suitable for small data sets.

Bayesian neural networks (BNNs, Bayesian NNs) also offer a probabilistic inter-

pretation of DL models by inferring distributions over the models' weights. The model offers robustness to over-fitting, uncertainty estimates, and can easily learn from small datasets [52]. As it is the case with Naive Bayes algorithm, Bayesian algorithms naturally incorporate a form of regularization (the prior), hence less prone to over-fitting the small dataset.

Another study, *Modern Neural Networks Generalize on Small Data Sets* [53], utilizes a linear program to decompose fitted neural networks into ensembles of low-bias sub-networks, which are found to be relatively uncorrelated. This process, similar to a random forest, leads to an internal regularization effect that contributes to the surprising resistance of neural networks to overfitting, even when trained on a small number of examples. In experiments that contain a much smaller number of training examples, deep neural nets are shown to achieve superior accuracy without overfitting. This study argues that building your networks deep enough would let you take advantage of this ensemble effect on small datasets.

In another study [54], there has been some success using stacked autoencoders to pre-train a network with more optimal starting weights, which helps avoiding local optima and other pitfalls of a bad initialization. The same study also claims that fully connected DNN that consists of 3 or more hidden layers shows its advantage over shallow neural network and support vector machine by achieving higher prediction accuracy and better generalization performance.

[55] empirically shows that, by issuing set-valued classifications, *Naive Credal Classifier-2* (NCC2) is able to isolate and properly deal with instances that are hard to classify in low data regimes (on which naive Bayes' accuracy considerably drops), and to perform as well as naive Bayes on the other instances.

In a nutshell, altering the architecture of the model can potentially improve its ability to learn from a limited amount of data.

3.3.4 Ensemble methods

Combining the predictions of multiple models, either by averaging or through a voting process, can often improve the overall performance of the model. This can be particularly useful in the case of small and largely unlabeled datasets, where

individual models may be prone to overfitting or underfitting.

Ensemble techniques [35], which involve aggregating the predictions of multiple models, can often result in improved accuracy and reduced variance compared to individual models. This can be achieved through various methods such as weighting the predictions of different models or using different values of hyperparameters for the same model. Such techniques can be particularly useful in situations where data is limited, as they can help mitigate overfitting and increase generalizability. It is also advisable to seek input from domain experts when implementing ensemble methods, as they may be able to provide valuable insights on the appropriate weighting of different models or the selection of suitable hyperparameters.

In ensemble learning, multiple models are combined to make a final prediction. Two common approaches in ensemble learning are bagging and boosting. Bagging (bootstrap aggregation) involves generating random samples of the training dataset with replacement, running a learning algorithm on each sample, and then taking the mean of all predictions. Boosting is an iterative method that adjusts the weight of each observation based on the previous classification. This approach aims to reduce bias error and build strong predictive models.

3.3.5 Transfer learning

If there are similar tasks or datasets for which labeled data is available, it may be possible to leverage this data to improve the performance of the model on the small and largely unlabeled dataset. This can be done through transfer learning, where a pre-trained model is fine-tuned on the new dataset. Transfer learning [30], also known as domain adaptation, is basically the leveraging the knowledge of a neural network learned by training on one task to apply it for another task (Figure 3.5). The learned weights of a model that was pre-trained on one dataset are used to 'bootstrap' training of early or all but the final layer of a modified version of the model applied to a different dataset.

In transfer learning, the knowledge gained from a source task is utilized to facilitate and accelerate the learning process for a new target task. This technique allows faster training, whereby the model just learns the weights of the last fully connected layers, then applies a low learning rate with finely tuned adjustment to the entire

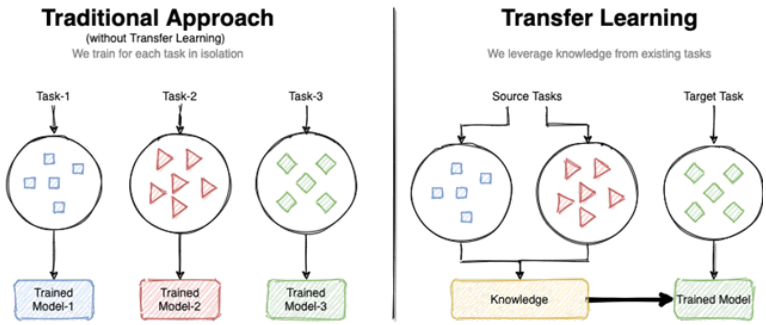


Figure 3.5: In traditional approaches, a separate model is created for each task, as illustrated in the figure on the left where three distinct models are used for three different tasks. In contrast, transfer learning utilizes knowledge acquired from source tasks to enhance the performance of the target task, as depicted in the figure on the right (source: [56]).

model's weights. This is achieved by two main concepts: Freezing and fine-tuning. It is a common strategy to freeze the layers in neural-networks that are supposed to be leveraged from the pre-trained model 'as is', and no weight updates are expected on these layers. Freezing the weights for the initial layers of the network usually gives better results on the smaller target set classes. [57] states that freezing the first two to three layers of features results in a significant performance boost over the baseline score, especially for smaller target set sizes under a thousand instances per class.

In transfer learning, it is important to ensure that the weights trained on the source task are relevant to the target task. If the weights are not relevant, transfer learning may not be effective. It has been observed that the benefit of a pre-trained network greatly decreases as the task the network was trained on diverges from the target task [58]. For example, if the source task involves classifying horses and zebras and the target task involves detecting benign and malign tumors, the weights from the source task may not be useful for the target task. In order to obtain good results, it is important to initialize the network with pre-trained weights that are relevant to the target task.

One of the key challenges in using transfer learning techniques is to ensure that the knowledge acquired from a source task is effectively transferred to a target

task, while avoiding negative transfer between tasks that are not closely related. Removing layers from a pre-trained model may also have negative effects, as it can alter the architecture of the model and potentially result in overfitting. It is therefore important to carefully consider the number of layers to include in the model.

In computer vision, deep neural networks trained on a large-scale image classification dataset such as ImageNet have proven to be excellent feature extractors for a broad range of visual tasks such as image classification and object detection. The sample illustration of transfer learning and details of a typical CNN architecture for image recognition can be seen at Figure 3.6.

Another important observation is that the performance of ImageNet architectures and the power of transferability across different datasets may differ given the type of the network (e.g., VGG-19, ResNet) and the nature of the dataset (see Figure 3.7 for more details).

There are different forms of transfer learning:

Inductive transfer learning refers to the use of knowledge learned from one task to improve the performance of another task with the same structure, but with different data. It is commonly used in real-world settings and is effective at improving the performance of the target task. This type of transfer learning may involve multitask learning, in which the model is trained on multiple related tasks, or self-taught learning, in which the model is trained on a large amount of unlabeled data and then fine-tuned on a small labeled dataset [56].

Inductive transfer learning is particularly useful when the target task has a limited number of labeled samples, as it allows the model to make use of the larger labeled dataset in the source domain (labeled data from both source and target domains are available for training). In order to induce the knowledge from the source domain, it is necessary to have labels available in the target domain. Inductive transfer learning is commonly used in real-world settings and is effective at improving the performance of the target task.

In a recent study, entitled *Simplified Transfer Learning for Chest Radiography*

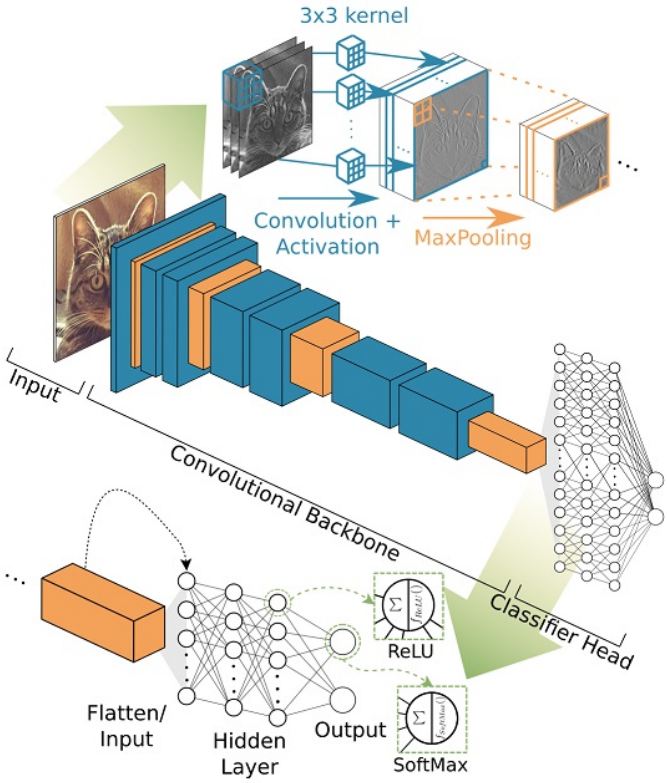


Figure 3.6: Schematic representation of a Convolutional Neural Network (CNN) architecture for image recognition [59]. The process begins with the input image, which undergoes a series of convolutional layers with applied activation functions, followed by max pooling layers to reduce dimensionality while retaining important features. This sequence forms the convolutional backbone, which is crucial for feature extraction. The extracted features are then flattened and fed into a dense hidden layer. The network concludes with a classifier head, consisting of additional dense layers with ReLU activation and a final SoftMax layer for output class probabilities. Each step in the architecture is designed to progressively abstract and condense the image information into a form that the model can use to make accurate classifications.

Models Using Less Data [61], the authors proposed an approach called supervised contrastive (SupCon), and compared with transfer learning from a nonmedical dataset. The models using the pretrained weights reduced label requirements up to 688-fold and improved the area under the receiver operating characteristic

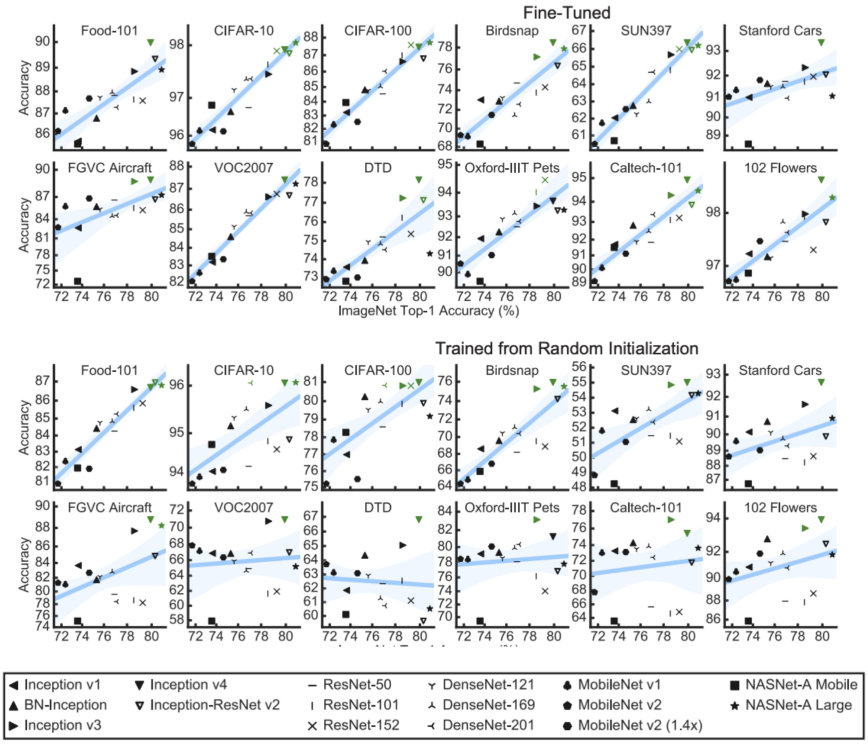


Figure 3.7: The performance of modern CNN architectures on popular datasets differ based on the nature of the datasets and whether the network is initialized from the ImageNet checkpoints (finetuned) or trained from scratch with random initialization (source: [60]).

curve (AUC) at matching dataset sizes. At the extreme low-data regimen, training small nonlinear models by using only 45 chest radiographs yielded an AUC of 0.95 (non-inferior to radiologist performance) in classifying microbiology-confirmed tuberculosis in external validation (see Figure 3.8 for more details).

Unsupervised transfer learning involves transferring knowledge between tasks that are similar, but with different data, and in which both the source and target tasks have unlabeled data. In some case, transfer learning without any labeled data from the target domain is also known as unsupervised transfer learning. Unsupervised transfer learning often involves techniques such as dimensionality

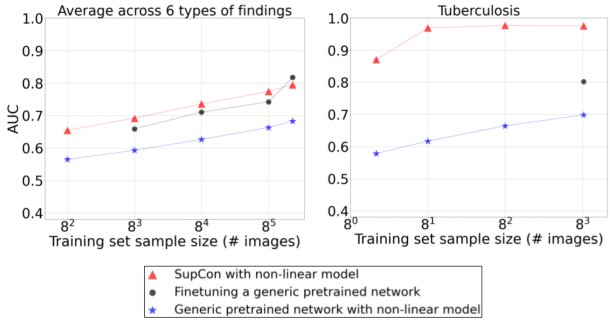


Figure 3.8: Efficacy of CXR (chest X-Ray) specific networks utilizing supervised contrastive (SupCon) learning compared to standard transfer learning from a non-medical dataset (red), with a control group using a generic pretrained network (blue). Performance is measured by the area under the receiver operating characteristic curve (AUC) for detecting various Chest X-Ray (CXR) abnormalities (left graph) and specifically tuberculosis (right graph). The SupCon approach, requiring significantly fewer labels, matches or surpasses the performance of the transfer learning model at equivalent dataset sizes, demonstrating up to a 688-fold reduction in label requirements. Notably, small nonlinear models trained on as few as 45 chest radiographs achieved an AUC of 0.95, comparable to radiologist assessment in identifying confirmed tuberculosis cases. Figures are taken from [61].

reduction and clustering to identify patterns in the data. In order to achieve that, we usually assume the existence of a common structure between source and target and leverage this information to perform the transfer [62].

Transductive transfer learning (also known as *domain adaptation*) refers to transferring knowledge between tasks with similar structure, but in which the source and target tasks have different data and the source task has a large amount of labeled data, while the target task has no labeled data (and usually following different distributions). In another term, the source and target domains have the same feature space while they can originate from different distributions. That is, the learning algorithm knows exactly on which examples it will be evaluated after training without any labelled samples in the target domain. This can become a great advantage to the algorithm, allowing it to shape its decision function to match and exploit the properties seen in the test set [63]. In this case, the model may use domain adaptation techniques to align the distribution discrepancy between domains in order to generalize the trained model to the domain of interest (adjust

to the different data distribution in the target task).

3.3.6 Parameter initialization

Initialization of neural network weights is an important factor that can impact the performance of a model during training. There are various methods for initializing weights, such as using constant values, sampling from a distribution, or using more sophisticated schemes like the so-called Xavier Initialization [64]. The authors of [64] demonstrated that networks initialized with Xavier achieved substantially quicker convergence and higher accuracy on the CIFAR-10 image classification task.

The choice of initialization method can affect the reproducibility and convergence speed of the neural network. The selection of initial values for the parameters of a neural network can significantly influence its performance. Inadequate initialization, such as random initialization of weights, may result in non-reproducibility and inferior performance, while initializing with constant values may delay convergence. Careful initialization can aid in learning with limited data, improve reproducibility, and optimize the training.

In a study comparing the effects of different weight initialization strategies on the performance of a CNN (Figure 3.9), it was found that initializing weights with values drawn from normal distributions with variances inversely proportional to the number of inputs into each neuron resulted in the best performance. The network achieved a validation accuracy of over 99% and a final loss two orders of magnitude smaller than networks initialized with weights set to zero (left plot on Figure 3.9) or drawn from normal distributions with a standard deviation of 0.4 (middle plot on Figure 3.9). These results suggest that careful weight initialization can significantly boost the convergence and performance of a neural network.

The effectiveness of transfer learning in dealing with small data problems can partly be attributed to parameter initialization since we basically try to pick the right initialization (initial checkpoints) pretrained on a similar but larger dataset. Hence we can state that the initial state of the parameters can significantly impact the optimization process, potentially leading to issues such as divergence, getting stuck at saddle points or local minima, and requiring a larger amount of training data

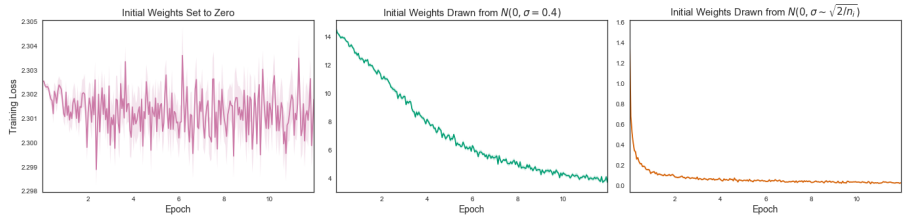


Figure 3.9: Comparative visualization of training loss trajectories using different weight initialization strategies for a CNN on the MNIST dataset. Each subplot displays the 10-batch rolling average of the loss incurred during the training of a basic CNN on the MNIST dataset, which consists of 60000 images of handwritten digits. The network was trained for a total of 12 epochs with a batch size of 128 images for each weight initialization strategy. The left plot shows the model with weights initialized to zero, exhibiting high variability and poor convergence. The middle plot illustrates weights initialized from a normal distribution with a standard deviation of 0.4, showing better, yet suboptimal performance. The right plot demonstrates the superior performance of weights initialized from a normal distribution with variance scaled inversely with the number of neuron inputs, yielding the most stable convergence and significantly lower loss. This strategy led to a network achieving over 99% validation accuracy and a notably smaller final loss compared to the other two methods, underscoring the importance of proper weight initialization in neural network training.(source: [65]).

for successful training. It is therefore important to carefully consider the initial parameter values chosen for a model.

3.3.7 Loss function reformulation

In modern DL research, the categorical cross-entropy loss after softmax activation is the method of choice for classification, that has not been questioned well enough. A recent paper, DL on Small Datasets without Pre-Training using Cosine Loss [66], claims to obtain around 30% increase in accuracy for small datasets when switching the loss function from categorical cross-entropy loss to a cosine loss (1 — cosine similarity) for classification problems.

The Cosine Loss function, which maximizes the cosine similarity between the output of a neural network and one-hot vectors [67] indicating the true class, has been found to be an effective method for learning from small datasets. This method has been shown to be superior to cross-entropy loss, possibly due to the inclusion of L2 normalization, which acts as a regularizer without the need for

additional hyperparameters. Experiments have demonstrated the effectiveness of this approach in small dataset scenarios (Figure 3.10).

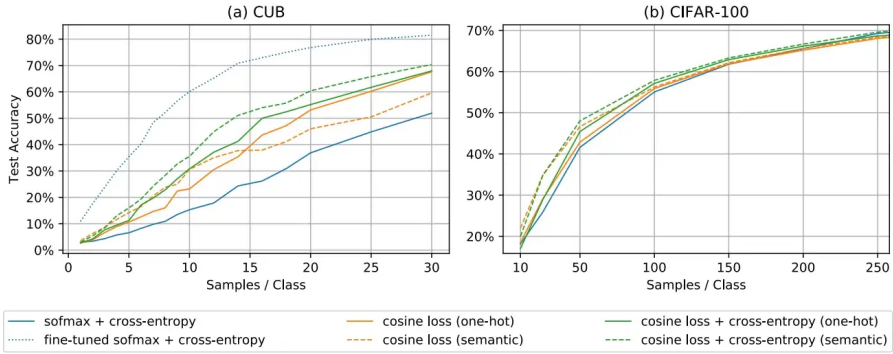


Figure 3.10: Test accuracy comparison between different loss functions on two datasets: (a) CUB (Caltech UCSD Birds) and (b) CIFAR-100. The graphs contrast the performance of softmax with cross-entropy loss and cosine loss with variants one-hot and semantic cross-entropy. The cosine loss, particularly with semantic cross-entropy, shows improved learning efficiency on the CUB dataset, a small dataset scenario, highlighting its advantage in scenarios with fewer samples per class. However, the performance gains are less distinct in larger datasets like CIFAR-100, indicating the loss function's relative impact based on dataset size. These findings suggest that cosine loss, due to L2 normalization acting as an implicit regularizer, is a robust choice for small datasets, outperforming the traditional cross-entropy loss in such contexts (source: [66]).

The Hinge loss function [68] has also been demonstrated to be effective in situations where resources are limited, allowing for the training of models on small datasets. In particular, the squared Hinge loss has been found to have faster convergence and improved performance compared to other variants of the Hinge loss. Additionally, it has been shown to be more resistant to noise in both the training set labels and the input space [69], [70].

3.3.8 Regularization techniques

Since the ML model trained with small data are prone to overfitting, regularization techniques can be considered an effective way of dealing with limited data in machine learning. Regularization is a technique used in ML to constrain the model fitting process and reduce the effective number of degrees of freedom without decreasing the actual number of parameters in the model. It is a popular method

to penalize overly complex models and prevent overfitting (penalizes the coefficients that cause the overfitting of the model), thus improving the generalizability of a model.

The penalty term is based on the magnitude of the model parameters, and its purpose is to discourage the model from fitting too closely to the training data. In the context of linear regression, regularization involves adding a penalty term to the cost function that is proportional to the magnitude of the coefficients. This forces the model to prefer solutions with smaller coefficients, which can help to mitigate overfitting and improve the model's ability to generalize to unseen data.

L1 Norm (Lasso) and **L2 Norm** (Ridge) regularization are two popular regularization techniques. In L1 regularization, the penalty term added to the cost function is the summation of absolute values of the coefficients, making the models with fewer non-zero parameters, which can be beneficial for the interpretability of model performance in a production setting. On the other hand, in L2 regularization, the penalty term added to the cost function is the summation of the squared value of coefficients, making the models with more conservative (closer to zero) parameters, which is similar to applying strong zero-centered priors to the parameters in a Bayesian framework. Generally, L2 regularization is more effective for improving prediction accuracy compared to L1 regularization.

Dropout [71] is a another method of regularization in which activations of randomly selected neurons are set to zero during the training process. This technique helps the network learn more robust features and reduces its reliance on the predictive power of a small group of neurons. [72] applied this concept to CNNs through the use of Spatial Dropout, in which entire feature maps are dropped out instead of individual neurons.

Weight decay (with AdamW - Adam with decoupled weight decay [73]), explained briefly in 3.3.12, is also one of the powerful techniques in regularization in which proper weight decay tuning help validation accuracy sometimes suddenly begins to increase from chance level toward perfect generalization, even long after severely overfitting on small datasets.

Batch normalization [74] can also be regarded as a regularization method that

normalizes the activations in a layer by subtracting the batch mean and dividing by the batch standard deviation. This technique is commonly used in the preprocessing of pixel values and has been demonstrated to improve the performance of CNNs. In addition to its regularization properties, batch normalization can also help stabilize the training process and reduce the sensitivity of the network to the choice of initialization parameters. [1] and [2] illustrated the impact of BN layer empirically under various settings and showed that the final BN layer, when placed before the softmax output layer, has a considerable impact in highly imbalanced image classification problems as well as undermines the role of the softmax outputs as an uncertainty measure. The impact of batch normalization on various settings will be explored in detail in Chapter 4.

3.3.9 Data augmentation

Data augmentation (DA) is a common technique used to virtually increase the size of a dataset by applying modifications to color, brightness, contrast, or adding noise to the existing data [75]. While this technique can be effective in improving the performance of ML models, it has the limitation of presenting the same samples in different forms to the model, which may not have a significant impact on generalization. Nevertheless, this can help to reduce overfitting. Due to its impact on by providing more stable and smoother response to various variations of input data, DA can even be considered as part of a broad set of regularization techniques aimed at improving model performance [76].

It is widely accepted in the field of ML that increasing the amount of data available, even if the quality is not optimal, can lead to improved model performance. Data augmentation can be effective because it incorporates prior knowledge into the dataset. The use of Generative Adversarial Networks (GANs) [77] and paired samples [78] are also being explored as methods for data augmentation.

DA techniques can be divided into two categories: data warping (basic image manipulations) and oversampling (Figure 3.11). Data warping techniques preserve the label of the original data and include transformations such as geometric and color manipulations, random erasing, adversarial training, and neural style transfer. Oversampling techniques, on the other hand, create synthetic instances and add

them to the training set. Examples of oversampling techniques include mixing images, feature space augmentations, and the use of GANs. It is worth noting that these categories are not mutually exclusive, and it is possible to combine techniques such as GAN sampling with data warping methods like random cropping to further increase the size of the training dataset [75].

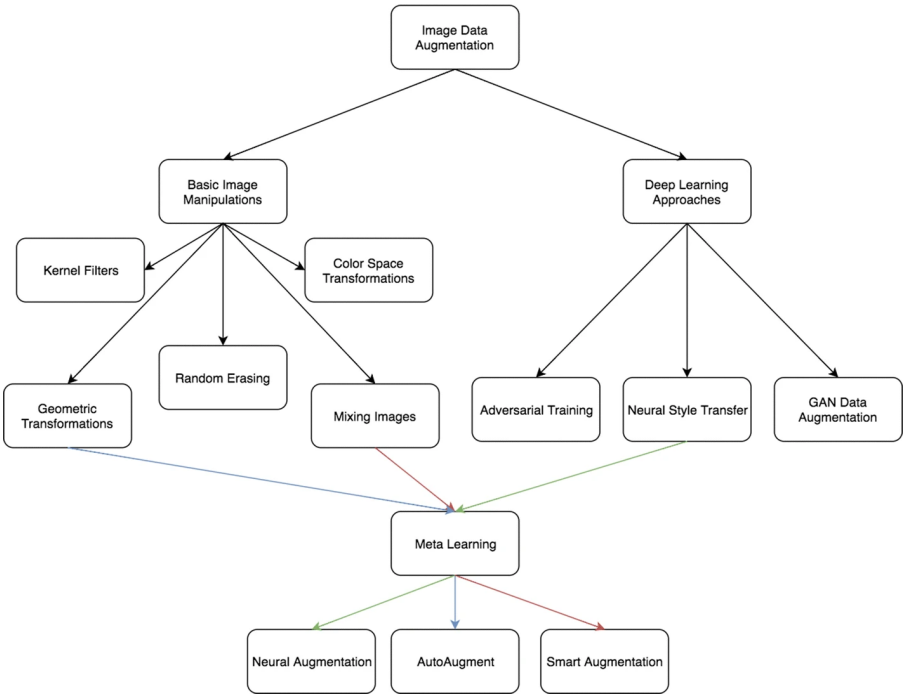


Figure 3.11: Image augmentation techniques (source: [75]).

GANs take random noise from a latent space and produce unique images that mimic the feature distribution of the original dataset. The model, based on image conditional GANs, takes data from a source domain and learns to take any data item and generalise it to generate other within-class data items. As this generative process does not depend on the classes themselves, it can be applied to novel unseen classes of data. There are a plethora of different types of GANs in the literature that can be used to generate synthetic data [79]. The comparison of basic image transformations and GAN-based augmentation can be seen at Figure 3.12.

Figure 3.12: Basic image transformations [80] and GAN-based augmentation methods [81].

Paired sample is a surprisingly effective data augmentation technique for image classification tasks. In this technique, called SamplePairing [78], a new sample from one image is synthesized by overlaying another image randomly chosen from the training data (i.e., taking an average of two images for each pixel). This simple data augmentation technique significantly improved classification accuracy in various benchmark datasets .

DA can bring several benefits to ML models. One of the main advantages is the improvement of model prediction accuracy. By increasing the amount of training data available, models can be more robust and less prone to overfitting. Moreover, data augmentation can help mitigate data scarcity, which can be particularly useful in scenarios where data collection and labeling are costly. Additionally, data augmentation can also help address class imbalance issues in classification tasks by generating synthetic samples that help balance the distribution of classes in the dataset. Overall, data augmentation can increase the generalization ability of models, leading to better performance on unseen data.

As a result, DA can be used to address certain types of biases present in a small dataset. However, it is not a comprehensive solution and cannot create new categories of data that are not already represented in the dataset. Nevertheless, it can be effective in mitigating biases related to factors such as lighting, occlusion, scale, and background. Additionally, DA can help to prevent overfitting by artificially increasing the size and diversity of the dataset, which can have characteristics similar to those of a larger dataset. This can be particularly useful when working with limited data, as it can allow for the development of more robust models.

With the majority of DA techniques being manually created, recently there are various method developed to automate DA processes such as AutoAugment [82] and Trivial Augment [83] that achieve significant improvements on benchmark datasets.

3.3.10 Synthetic data generation

Synthetic data refers to artificially generated data that is designed to mimic real-world data. It can be used in a variety of contexts to preserve privacy in sensitive domains, such as medical and transactional data. Synthetic data can also be generated using a small amount of well-labeled data, and there are various techniques, such as SMOTE (Synthetic Minority Over-sampling Technique) [84], ADASYN (Adaptive Synthetic Sampling Approach) [85], Variational AutoEncoders (VAEs) [86], and the aforementioned GANs, that can be used to generate synthetic data. The use of synthetic data can facilitate the processing of large amounts of real-world data and accelerate the time and energy required for such processes.

SMOTE: One approach to addressing the issue of imbalanced data is generating synthetic data, which can be accomplished through techniques such as Synthetic Minority Over-sampling Technique (SMOTE) and Modified-SMOTE [87]. These methods generate new data points by taking the minority class data points and creating new points that lie between two nearest data points joined by a straight line in the feature space. The number of nearest neighbors used for data generation can be adjusted as a hyper-parameter based on the requirements of the problem. However, it is important to note that generating synthetic data can increase the risk of overfitting due to the presence of duplicate data.

One of the advantages of using SMOTE is that it mitigates the problem of overfitting caused by random oversampling, as synthetic examples are generated rather than replications of instances. Additionally, SMOTE does not result in the loss of useful information. However, a disadvantage of using SMOTE is that it does not take into consideration neighboring examples from other classes, which can result in an increase in overlapping of classes and the introduction of additional noise. Additionally, SMOTE may not be effective for high dimensional data (i.e. a dataset with large number of attributes or features). According to [88], high-dimensional problems are problems where the number of targeted features p is much larger

than the number of observations N, often written $p \gg N$.

The M-SMOTE algorithm is a modified version of the SMOTE method [87], which takes into account the underlying distribution of the minority class when generating synthetic data. This approach involves classifying minority class samples into three categories: security or safe samples, border samples, and latent noise samples. Security samples refer to those data points that have the potential to improve the performance of a classifier, while noise samples are those that may decrease the performance of the classifier. Data points that are difficult to categorize into either category are referred to as border samples. These categories are determined by calculating the distances between minority class samples and the training data. M-SMOTE then randomly selects a data point from the k nearest neighbors for security samples, selects the nearest neighbor for border samples, and does not generate synthetic data for latent noise samples.

Generative Adversarial Networks (GANs): Generative adversarial networks (GANs) are a class of neural networks [89] that consist of two sub-networks: a generator and a discriminator. The generator is responsible for synthesizing output data, while the discriminator is responsible for distinguishing between the synthesized data and real data. The generator and discriminator are trained to compete against each other, with the generator attempting to generate outputs that are indistinguishable from real data, and the discriminator trying to accurately identify whether an output is real or synthesized. Through this process, GANs are able to produce outputs that are highly realistic in appearance.

The use of GANs in data augmentation has received considerable attention due to their ability to generate new training data that leads to improved classification model performance. For instance, [90] used a large CT image database and trained a GAN to transform contrast CT images into non-contrast images; and then used the trained model to augment their training using these synthetic non-contrast images, and managed to reduce manual segmentation effort and cost in CT (computer tomograpght) imaging. In another study, [91] utilized public datasets and created synthetic versions using various GAN models. The effectiveness of these synthetic datasets as training data was evaluated through two methods. First, by comparing the accuracy, precision, and recall of a decision tree classifier trained on the original

data and one trained on the synthetic data. Surprisingly, in some instances, the classifier trained on synthetic data performed better than the one trained on the original data, indicating that GAN-based data augmentation can be a useful strategy to prevent overfitting. [92] applied GAN to approximate the true data distribution and generate data for the minority class of various imbalanced datasets and compared the performance of GAN with multiple standard oversampling algorithms.

[92] also argued that traditional methods such as zooming, cropping, and rotating are effective for object classification but are not applicable in cases like time series data presented in images where there is no singular object to classify. To enhance the classification accuracy through the deep neural network's capability to handle intricate data, the authors proposed a GAN-CNN classifier combination that showed a superior accuracy compared to the other ML methods [93].

The original GAN architecture, which utilizes multilayer perceptron networks in the generator and discriminator networks, is able to generate acceptable images for simple datasets like MNIST handwritten digits, but is not effective for producing high quality results for more complex, high resolution datasets. There have been numerous studies that have modified the GAN framework, including through the use of alternative network architectures, loss functions, and evolutionary methods, among others. These modifications have led to improvements in the quality of samples generated by GANs. Several new GAN architectures, including DCGANs [94], StackGAN [95], Progressively-Growing GANs [96], CycleGANs [97], and Conditional GANs [98], have been proposed and have been found to have potential applications in data augmentation.

3.3.11 Problem reduction

Problem reduction refers to the process of transforming a new or unknown problem into a known problem that can be easily solved using existing techniques. This approach has been demonstrated to be effective in scenarios where limited datasets are available. For instance, sound classification problem can be transformed into image classification by converting the voice clips into images (spectograms), which can then be addressed using state-of-the-art computer vision architectures and

techniques such as transfer learning. Research has shown that this approach can produce satisfactory results even with small datasets [99].

Another example would be transforming the image classification problem into two-stage problem such as object detection to extract patches and then image classification problem to classify the patches into target classes. This approach can be highly effective when there is not enough labelled images for the classification problems. Imagine we are trying to detect if a solar panel array (a group of solar panels that are connected together, collectively converting solar radiation into electricity) is defect or not; and the dataset is composed of a list of solar panel arrays each of which has only one class. Using problem reduction approach, rather than classifying the entire array into a single class, we can do the followings:

1. Crop the panels from the array using image processing and segmentation algorithms (Figure 3.13).

2. Using the original defect coordinates, assign positive defect label to the panel that contains these coordinates; and assign negative class (no defect) to all the other panels (now we have i, the number of defected panels, as the positively labelled panels, versus $(n - i)$, the complementary number of panels in an array, as the negatively labelled panels) (Figure 3.14).

3. Train a classifier on this dataset to find out defect panels.

4. In inference time, run segmentation algorithm at first to isolate the panels and then crop them.

5. Send each segmented panel into the classifier to detect if a panel has defect or not.

Another advantage of using this approach is reducing the problem complexity by filtering out the irrelevant portions of an image and only showing the useful parts. The learning capacity of a model trained on cropped patches is usually much higher compared to the one trained with entire images having many irrelevant parts and background noises.

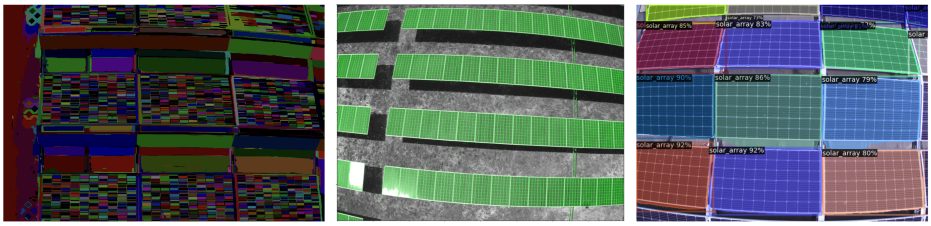


Figure 3.13: Segmenting and cropping the panels from solar array. Left: Global Contrast based Salient Region Detection (SGD) [100], middle: Enhanced line segment drawing (ELSED) [101], right: Detecting twenty-thousand classes using image-level supervision (DETIC) [102]. Images are taken from NORCE-PV dataset mentioned in [3].

3.3.12 Optimization techniques

The optimization of neural networks using gradient descent algorithms is significantly impacted by the size of the training dataset. While using smaller training sets, such as in the case of stochastic gradient descent, can result in faster training processes, updates may exhibit larger fluctuations. Conversely, training with larger datasets can be computationally expensive and slow. Therefore, the selection of the appropriate training dataset size is an important consideration in the optimization of neural networks.

Gradient descent (sometimes called batch gradient descent) is an example of an algorithm that performs better (in terms of computation speed) when the dataset is small, because it computes the gradients on the whole dataset in each iteration, as opposed to stochastic gradient descent which uses only a small part of the data in each iteration.

In the context of few-labelled samples, gradient-based optimization algorithms have been shown to be less effective due to two main reasons. Firstly, these optimization algorithms, such as momentum, AdaGrad, AdaDelta, and ADAM, are not specifically designed to perform well under the constraint of a limited number of updates. When applied to non-convex optimization problems, these algorithms do not provide strong guarantees of convergence speed, and may require a large number of iterations to reach a good solution. Secondly, the network must be

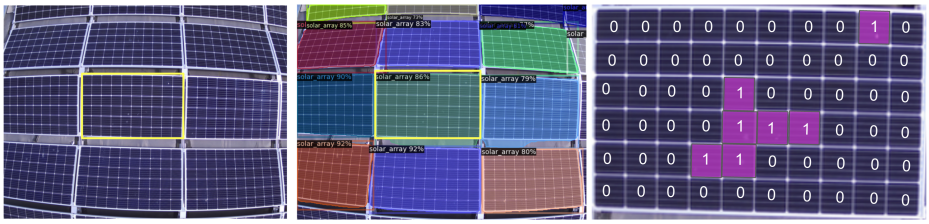


Figure 3.14: The solar panel array on the left image has 9 visible panels and only one of them is labelled as a defect. Rather than using this entire image in a 'solar array classification task,, segmenting & cropping the relevant panel at first (middle image), then each cell within a panel and then assigning a defect status labels ($0 = no_defect, 1 = defect$) for each cell (image on the right), and then sending these $9x60 = 540$ patches to 'solar cell classification' task would result in a better performance as we will be ending up more data points to help the model learn better. The image on the right indicates the solar panel marked with yellow bounding box on the other images (left and middle). The original image is taken from NORCE-PV dataset mentioned in [3].

initialized with random parameters for each separate dataset, which hinders its ability to converge to a good solution after a few updates. Overall, these factors make it difficult for gradient-based optimization to effectively learn from small datasets [103].

In another study [37] that investigates the impact of optimization methods that work well under small data regimes, the authors show that long after severely overfitting, validation accuracy could suddenly begin to increase from random level toward perfect generalization (see Figure 3.4).

Normally, when using a supervised learning approach, reducing the amount of training data leads to a decrease in the model's ability to generalize when the optimization process is able to fit the training data perfectly. However, [37] found a different outcome: the model's performance remains constant at 100% within a certain range of training dataset sizes, but the time needed for optimization increases rapidly as the dataset size is reduced. They show empirically that different optimization algorithms lead to different amounts of generalization and the amount of optimization required for generalization quickly increases as the dataset size decreases. They compare various optimization details to measure their impact on data efficiency and find that *weight decay with AdamW* (Adam with decoupled

weight decay [73]) is particularly effective at improving generalization on the tasks they studied (see Figure 3.15). [73] illustrates that *Adam* generalizes substantially better with decoupled weight decay than with L2 regularization, achieving 15% relative improvement in test error on several image benchmark datasets.

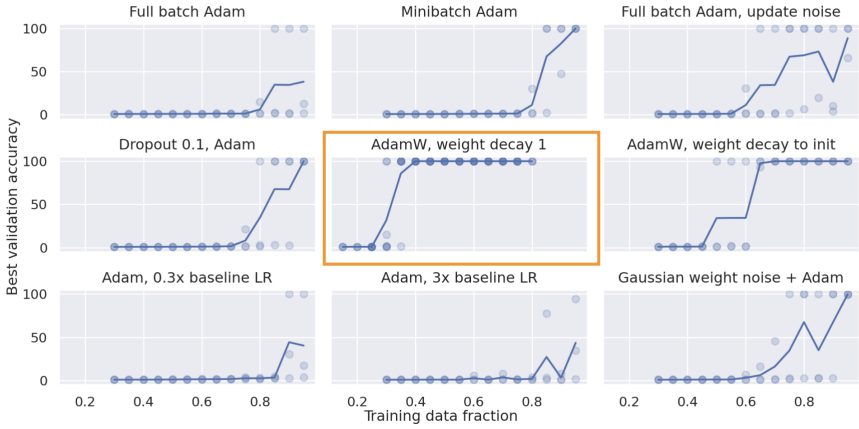


Figure 3.15: Different optimization algorithms lead to different amounts of generalization within an optimization budget of 10^5 steps. Weight decay (with AdamW) improves generalization the most, but some generalization happens even with full batch optimizers and models without weight or activation noise at high percentages of training data. Suboptimal choice hyperparameters severely limit generalization (source: [37]).

In another study [104] investigating the best hyperparameter settings to couple self-supervised learning paradigms with semi-supervised techniques in low data regimes, the authors empirically demonstrate that weight decay is usually among the most important hyperparameters to tune while training deep neural networks, no matter what the size of the dataset is as depicted in Figure 3.16.

The "hypersweep curves" in Figure 3.16 depict the performance of a supervised baseline model on a fraction of the ILSVRC-2012 dataset, illustrating the impact of varying hyperparameters on model accuracy. The curves represent the sorting of models by accuracy with a fixed hyperparameter. For both figures, the rightmost point on each curve indicates the peak performance for that hyperparameter value. The spread between curves at these points reflects the sensitivity of the model to the hyperparameter, while the overall shape and proximity of the curves suggest

the robustness and interdependence of the hyperparameter values. The chart on the left uses the full custom validation set, whereas the other chart employs a reduced validation set with one image per class, which is shown to be sufficient for hyperparameter evaluation. Notably, the experiments reveal that weight decay and the number of training epochs significantly influence model training on limited data. Contrary to common belief, reducing model capacity by decreasing depth or width does not enhance performance; instead, deeper and wider models demonstrate superior performance and robustness, even with limited training data. These findings challenge prevailing assumptions and support recent observations that wider models can facilitate optimization.

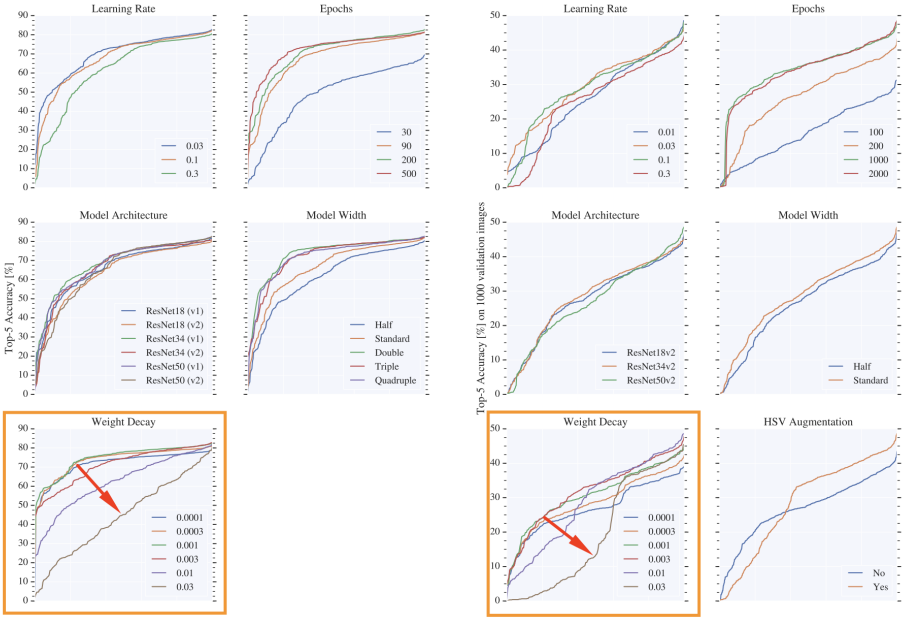


Figure 3.16: Performance impact of hyperparameter variation on a baseline model trained with a subset (10% on the left and 1% on the right) of the ILSVRC-2012 dataset. The curves demonstrate model accuracy against fixed hyperparameter values, with the largest spread between curves illustrating the pronounced effect of weight decay on model performance. These results underscore the significance of weight decay in training with limited data and challenge the notion that reduced model capacity benefits small dataset performance as well as indicating that weight decay is usually among the most important hyperparameters to tune while training deep neural networks as it may give the largest boost in validation accuracy compared to other hyperparameters (source: [104]).

3.3.13 Using physics-informed neural networks

The incorporating existing physical principles into ML can be a useful approach for modeling small data problems and developing more powerful models that learn from data and build upon our existing scientific knowledge. With the growth of ML and the availability of large amounts of scientific data, data-driven approaches have become increasingly prevalent in scientific research, including information science, mathematics, medical science, materials science, geoscience, life science, physics, and chemistry [105]. These approaches, also known as scientific ML (SciML) [106], do not require an existing theory and allow for the use of ML algorithms to analyze scientific problems solely based on data. This shift in the scientific method represents a departure from the traditional method of designing a well-defined theory and refining it through experimentation and analysis to make new predictions.

The physics-informed neural network (PINN) is a deep learning method that bridges the gap between machine learning and scientific computing by incorporating physical principles into ML. PINN has superior approximation and generalization capabilities, which made it gain popularity in solving high-dimensional partial differential equations (PDEs) [107], and has been used in various applications such as weather modeling, healthcare, and manufacturing.

PINN is able to predict the solution far away from the experimental data points, and thus performs much better than the naive network. PINNs can be easily applied to many other types of differential equations too, and are a general-purpose tool for incorporating physics into machine learning. The idea is adding the known differential equations directly into the loss function when training the neural network. This additional "physics loss" in the loss function tries to ensure that the solution learned by the network is consistent with the known physics [108].

Similarly, [109] also used a deep neural network to learn solutions of the wave equation, using the wave equation and a boundary condition as direct constraints in the loss function when training the network. By using the physics constraint in the loss function the network is able to solve for the wavefield far outside of its boundary training data, offering a way to reduce the generalisation issues of

existing DL approaches.

Figure 3.17 shows the effect of incorporating the harmonic oscillator formula into a scientific task to find a model that is able to accurately predict new experimental measurements given the first few measurements (observations). The problem is, using a purely data-driven approach (looking only at the actual values of the unknown physical process) cannot generalize beyond the observations. Once we incorporate the underlying physics formula into loss function, the model is able to generalize further beyond training dataset without using additional data points.

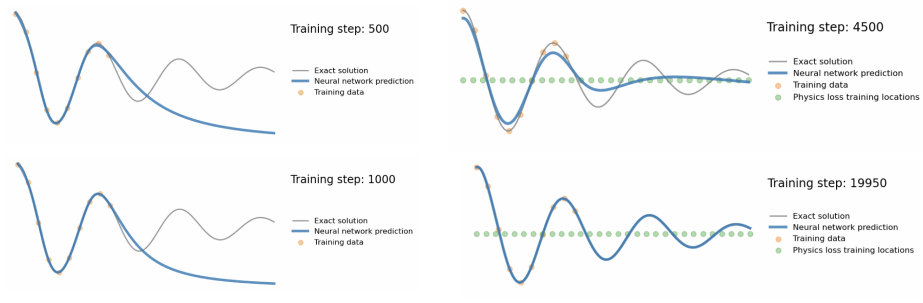


Figure 3.17: The physics-informed neural network is able to predict the solution far away from the experimental data points, and thus performs much better than the naive network. On the left hand side, even though the neural network can accurately model the physical process within the vicinity of the experimental data, it fails to generalise away from this training data. Once we incorporate the underlying physics formula into loss function, on the right hand side, the physics-informed network is able to generalize further beyond training dataset without using additional data points. Figures are taken from [110].

In short, PINN is a deep learning method that bridges the gap between machine learning and scientific computing by incorporating physical principles into ML. PINN has superior approximation and generalization capabilities, which made it gain popularity in solving high-dimensional partial differential equations (PDEs), and has been used in various applications such as weather modeling, healthcare, and manufacturing.

3.3.14 Unsupervised learning techniques

Unsupervised learning is a ML approach that aims to learn patterns in data without the use of labels. It relies on the model learning the structure of the input data based on the features present in the data. Feature extraction, which involves identifying and extracting relevant characteristics from the raw data, can often improve the performance of unsupervised learning approaches [111].

Common techniques used in unsupervised learning include clustering, which groups data points based on shared features, anomaly detection, which identifies data points that do not fit the distribution of existing examples, and manifold learning, a neural network approach that reduces the complexity of the data by learning latent variables such as Principal Component Analysis (PCA). Unsupervised learning can be applied to various types of data, including numerical and categorical data.

In the absence of labeled data, unsupervised learning techniques can be used to try to extract useful information from the dataset. For example, clustering algorithms can be used to group similar examples together, and dimensionality reduction techniques can be used to identify patterns and relationships within the data.

3.3.15 Semi-supervised learning

As the name suggests, Semi-supervised learning (SeSL) lies between the two extremes of supervised and unsupervised learning in terms of the availability of labeled data. In SeSL, the task is performed by utilizing both labeled and unlabeled datasets, with the aim of gaining a deeper understanding of the underlying data structure. Typically, SeSL is performed using a small labeled dataset and a relatively larger unlabeled dataset. The ultimate goal of this approach is to develop a predictor that can accurately predict future test data, surpassing the performance of predictors learned from labeled training data alone [112].

The application of SeSL techniques allows for the leveraging of labeled data, while also deriving structure from unlabeled data to improve the overall performance of the task. This is particularly beneficial in scenarios where the size of the labeled dataset is small. In such cases, traditional supervised learning algorithms are often prone to overfitting. However, by incorporating the understanding of structure

derived from unlabeled data during the training process, SSL effectively alleviates this issue. Additionally, SeSL methods provide an alternative solution to the challenge of building large labeled datasets for learning a task. These techniques are a step closer to mimicking the way humans learn, providing a more efficient and effective approach.

In recent literature, it has been shown that popular approaches to SeSL involve the introduction of a new loss term during training in a typical supervised learning setting. There are three main concepts that are typically used to achieve SeSL, namely Consistency Regularization, Entropy Minimization and Pseudo Labeling that allow SeSL to improve the performance of a supervised learning task when dealing with limited labeled data. Additionally, using generative models and graph-based methods in semi-supervised learning can be found in [113].

Consistency Regularization is a technique that aims to train a model that is robust to various data augmentations [114]. It enforces that the output of the predictor should not significantly change when realistic perturbations are made to the data points. This is achieved by minimizing the difference between the prediction on the original input and the prediction on the perturbed version of that input (decreasing the distances between features from differently augmented images). The idea behind this approach is to leverage the unlabeled data to find a smooth manifold on which the dataset lies. By achieving invariance to various data augmentations, the model can achieve robustness and generalize better to unseen data.

The most notable examples of Consistency Regularization include temporal ensembling (pi-model) [115], mean teachers (weight-averaged consistency targets) [116] and virtual adversarial training (VAT) [117] methods. Recently, Fan et al. [114] proposed a new perspective on the concept of consistency regularization, suggesting that instead of training a model that is invariant to all types of augmentations, it could be more beneficial to focus on improving equivariance on strongly augmented images, a simple yet effective technique, known as Feature Distance Loss (FeatDistLoss) that is designed to improve data-augmentation-based consistency regularization.

Entropy minimization [118] aims to encourage more confident predictions on unlabeled data. The model is trained to have low entropy predictions, regardless of the ground truth. Additionally, the confidence for all classes for an input example should sum to 1. This objective encourages the model to give high confidence predictions, and discourage the decision boundary from passing near data points where it would otherwise be forced to produce a low-confidence prediction.

The field of learning theory has traditionally focused on the two extremes of the statistical paradigm: parametric statistics, where examples are known to be generated from a known class of distribution, and distribution-free Probably Approximately Correct (PAC) [27] frameworks. However, Semi-supervised learning, which involves the use of both labeled and unlabeled data, does not fit neatly into these frameworks. This is because the usefulness of unlabeled data depends on the underlying distribution of the data, and no positive statement can be made without making distributional assumptions since unlabeled data coming from some distributions might be non-informative. This means that generalizing from labeled and unlabeled data may differ from transductive inference [118].

In this regard, the entropy minimization framework proposes an estimation principle applicable to any probabilistic classifier, aiming at making the most of unlabeled data when they are beneficial, while providing a control on their contribution to provide robustness to the learning scheme. This framework also shows that *unlabeled examples are mostly beneficial when classes have small overlap*, and provides a means to control the weight of unlabeled examples, and thus to depart from optimism when unlabeled data tend to hamper classification.

Pseudo-Labeling is a simple yet effective approach to achieve SeSL, and also known as proxy-labelling, self-training or co-training [119]. The method involves training a model on a labeled dataset, then using it (and based on some heuristics) to make predictions on unlabeled data. The examples from the unlabeled dataset where the model's prediction is confident (above a predefined threshold) are then selected and the predictions are considered as pseudo-labels. This pseudo-labeled dataset is then added to the original labeled dataset and the model is retrained on the expanded labeled dataset. This process can be repeated multiple times. Some examples of such methods are Self-training, Co-training and Multi-View Learning

[120].

Pseudo-Labeling is closely related to self-training, where the model is trained using its own predictions as labels. This approach allows for the efficient use of unlabeled data, as it can boost the performance of the model without the need for manual annotation. This technique is easy to implement and can be used in various applications where labeled data is scarce.

Recently, SeSL algorithms based on deep neural networks have been successful in achieving good results on standard benchmark tasks. However, it is argued that these benchmarks do not fully reflect the challenges that SeSL algorithms would face in real-world applications. In order to address this, [121] created a unified re-implementation of various widely-used SeSL techniques and tested in a suite of experiments. The results of these experiments indicate that the performance of simple baselines which do not use unlabeled data is often under-reported in the literature (see Figure 3.18).

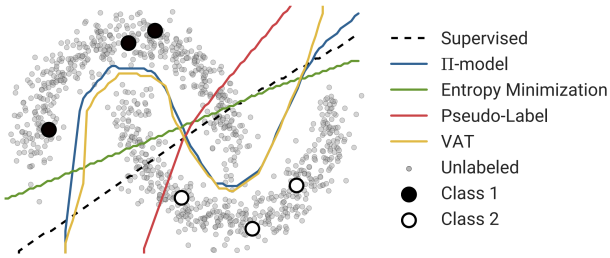


Figure 3.18: In the "two-moons" dataset, virtual adversarial training (VAT) and temporal ensembling (pi-model) methods were able to learn a highly accurate decision boundary with a relatively small amount of labeled data, specifically 6 data points depicted as large white and black circles in this figure. In contrast, Pseudo-Labeling was found to be inadequate in this context, resulting in the learning of a linear decision boundary. This highlights the limitations of Pseudo-Labeling as compared to VAT and pi-model in this specific dataset (source: [121]).

Additionally, the results show that different SeSL methods have varying sensitivity to the amount of labeled and unlabeled data. Moreover, the performance of these methods can degrade substantially when the unlabeled dataset contains examples that are out-of-distribution. These findings highlight the importance of rigorous

testing and evaluation of SeSL algorithms in real-world scenarios, in order to fully understand their capabilities and limitations [121].

In another study, [122] propose a new SeSL method called DP-SSL that adopts an innovative data programming (DP) scheme to generate probabilistic labels for unlabeled data. Different from existing DP methods that rely on human experts to provide initial labeling functions (LFs), they developed a multiple-choice learning (MCL) based approach to automatically generate LFs from scratch in SeSL style.

As a a result, it is essential to exercise caution when utilizing the technique of pseudo-labeling, as it has been observed that the model predictions can be fallacious at times. Furthermore, the model may generate several erroneous predictions for unlabeled data, leading to a detrimental feedback loop that may exacerbate the deterioration of performance. The same study also suggests that SeSL could be preferred by practitioners when there are no high-quality labeled datasets, labeled data is collected by sampling i.i.d. from the pool of unlabeled data and when the labeled dataset is large enough to accurately estimate validation accuracy.

For a detailed and comprehensive review of SeSL, the reader is referred to the Semi-Supervised Learning Book [112].

3.3.16 Self-supervised learning

Self-supervised learning (SSL) is a subcategory of unsupervised learning that utilizes unlabeled data to learn the representation of the data. This process is accomplished through a three-step process: generating input data and labels from the unlabeled data based on an understanding of the data, pre-training the model with the generated data and labels, and fine-tuning the pre-trained model for specific tasks of interest. In SSL, the objective is to learn generalizable and robust representations that can be applied to a variety of tasks and datasets, rather than simply learning high-level features.

Through self-supervised learning, it is possible to generate pre-trained backbone networks to extract features from domain-specific images and convert them into numerical vectors known as embeddings. This allows for the creation of models with fewer data and computational resources within that particular domain. In

some cases, even with less labelled data, this approach has enabled performance comparable to state-of-the-art DL models across various prediction tasks.

The most salient similarity between self-supervised and semi-supervised learning is that both approaches do not rely entirely on manually labeled data. However, this is where the similarity ends in broader terms. Self-supervised learning relies on the inherent structure of the data to make predictions, and does not require any labeled data (the system learns to predict part of its input from other parts of it). In contrast, semi-supervised learning still utilizes a limited amount of labeled data to guide the model's learning process.

Self-supervised learning is a primary component of this dissertation and will be elaborated in detail in Chapter 5.

3.3.17 Zero-shot, one-shot and few-shot learning

Zero-shot learning (ZSL) is a ML method in which a model is able to complete a task (classify and predict the class of an unseen sample, or detect an unseen object) without having received any training examples. It is a variant of transfer learning, where the model is trained on a set of classes and is able to generalize to unseen classes by leveraging additional knowledge about those classes, such as their textual descriptions or attributes. Unlike transfer learning, ZSL involves transferring knowledge between disparate feature and label spaces.

ZSL can be particularly useful when the number of classes in a dataset is large and obtaining labeled data for all classes is infeasible or impractical. It has garnered significant attention in recent years due to its ability to perform classification tasks with limited annotated data. This approach has been applied to various domains, including healthcare for medical imaging and COVID-19 diagnosis using chest x-rays, as well as unknown object detection in autonomous vehicles. As research continues to focus on methods that utilize minimal data with minimal annotation, the potential applications for zero-shot learning will likely expand.

In ZSL, when no labeled instances of the target classes are present in the training data, auxiliary information is utilized to facilitate the classification of unseen classes. This auxiliary information may take the form of class descriptions, known

attributes, semantic information, or word embeddings, and is necessary in order to effectively solve the ZSL problem. Through this process, ZSL techniques are intended to learn intermediate semantic layers and their properties, then apply them to predict a new class of data at inference time. For example, if we have a model that has been trained on images of horses and is presented with an image of a zebra, it might be able to recognize the zebra as a novel class based on its similarity to the known class of horses and the additional information that zebras have black and white stripes.

In ZSL, a labeled training set of seen classes and knowledge about the semantic relationships between seen and unseen classes is used to classify instances belonging to unseen classes. This is achieved through the use of a high-dimensional vector space, called the semantic space, in which the knowledge from seen classes can be transferred to unseen classes. The process of ZSL typically involves learning a joint embedding space in which both semantic vectors (prototypes) and visual feature vectors can be projected, and using nearest neighbor search to match the projection of an image feature vector with that of an unseen class prototype in this space [123].

Few-shot learning, also known as low-shot learning, on the other hand, is a method for classification when there are only a few examples per class. It is typically implemented in the form of N-way k-shot problems, where there are N classes and k labeled examples for each class. This approach utilizes meta-learning across a range of classification tasks in order to quickly adapt to a new task. The meta-learning model is trained on a large set of N-way k-shot tasks drawn from a similar dataset. Few-shot learning is simply an extension of ZSL, but with a few examples to further train the model.

Similarly, one-shot learning involves the use of only one instance or example of data for training, as humans have the ability to learn even with a single example and are still able to distinguish new objects with high precision. This approach is particularly useful in tasks such as identifying an image of a person in identification documents, where a large dataset may not be available. One approach to address this challenge is to modify the loss function to be more sensitive to subtle differences in the data and learn a better representation of it. Siamese networks are a commonly

used method for image verification in one-shot learning.

3.3.18 Meta learning

Meta-learning, also known as learning to learn, is a ML approach that aims to learn how a ML model learns and use this knowledge to improve the training process. By being trained on a variety of similar tasks, a meta-learner can learn the most effective way to learn an unseen task. Meta-learning has the potential to be particularly useful for limited data problems, as it can enable the rapid adaptation to a new environment and generalization to unseen tasks with only a few examples. This is achieved by learning a high-level representation of the learning process itself, allowing the model to adapt to new tasks more efficiently.

Meta learning algorithms can achieve superior generalization performance compared to non-meta learning algorithms, even in the presence of small data. However, meta-learning requires a large number of tasks from a similar dataset for meta-training. Once the meta-model is trained, it can be applied to learn an unseen task with a small amount of data, and can also support lifelong learning by continuously improving as it is being used.

In meta-learning, the aim is to learn the learning process itself, rather than simply recognizing patterns in training data and generalizing to unseen data, as is the goal in conventional supervised learning. During the meta-training phase, the algorithm learns to identify similarities and differences between examples from different classes in the training set. In each iteration, a support set containing n labeled examples from k classes is used along with a query set consisting of previously unseen examples from unknown classes. During training, the loss function assesses the performance on the query set, based on knowledge gained from its support set and will backpropagate through these errors.

One of the most successful meta-learning algorithms that is designed to work well with limited data is known as Model-Agnostic Meta-Learning (MAML). The key idea underlying this method is to train the model's initial parameters such that the model has maximal performance on a new task after the parameters have been updated through one or more gradient steps computed with a small amount of data from that new task [124].

Meta-learning can be conceptualized as an optimization problem, where a second network is used to predict the parameters for a given task, or where an initialization for a neural network is learned. Additionally, meta-learning can be approached through the use of similarity among features in examples, by embedding these examples into a vector space and using a metric for classification. Finally, meta-learning can also be achieved through the incorporation of prior knowledge about the structure of naturally occurring tasks, in order to improve model performance.

3.3.19 Harnessing model uncertainty

In ML and statistics, traditional methods for modeling uncertainty rely on probability theory. However, it has been argued that conventional approaches to probabilistic modeling, which capture knowledge in terms of a single probability distribution, fail to distinguish between two distinct sources of uncertainty, referred to as *aleatoric* and *epistemic* uncertainty. Aleatoric uncertainty, also known as statistical uncertainty, refers to the notion of randomness, or the variability in the outcome of an experiment that is due to inherently random effects. An example of this type of uncertainty is coin flipping, where the data-generating process has a stochastic component that cannot be reduced by any additional source of information. In contrast, epistemic uncertainty, also known as systematic uncertainty, refers to uncertainty caused by a lack of knowledge (about the best model). This type of uncertainty is caused by ignorance of the agent or decision maker, and can in principle be reduced on the basis of additional information [125]. Knowing the type of uncertainty during model development is a crucial component and helps the modeler avoid seeking more data if it is an aleatoric uncertainty in which the uncertainty is irreducible even if more data is provided.

In other words, epistemic uncertainty refers to the reducible part of the total uncertainty, whereas aleatoric uncertainty refers to the irreducible part. Recognizing and distinguishing between these two types of uncertainty allows for more nuanced and accurate modeling of uncertainty in a wide range of applications. In some cases, both aleatoric and epistemic uncertainties might be present, but it can be challenging to determine which type of uncertainty should be associated with a specific phenomenon during the modeling phase. This distinction is important as it affects the way we model the uncertainty and model, while it's not always

well defined. In some cases, it may be necessary to gather more data or perform additional experiments to accurately categorize the source of uncertainty [126].

When the sample size is small, specifying the right model class is not an easy task, so the model uncertainty will typically be high due to the lack of evidence in favor of any class (an inherent problem especially with semi-supervised learning wherein we need to define a threshold that decides to which class a data point belongs).

In an ideal scenario, even with powerful models that can generalize effectively beyond test sets, it is expected that a degree of uncertainty would exist for samples that are entirely unknown. This uncertainty can be leveraged to estimate areas in which the model may not have sufficient knowledge. This is particularly important in low data regimes, where it is plausible that not all parts of the distribution are equally represented. In such cases, it is important that the model's predictions for unseen samples return a degree of uncertainty that can be utilized for various purposes, such as reducing false positive rates, identifying predictions that require human intervention, or as a threshold in semi-supervised or active learning approaches.

Thus, it is important to understand and model the uncertainty present in the problem in order to know when the model's predictions can be trusted. One important aspect of this is considering the confidence of the predictions made by the model in addition to the predictions themselves. To achieve this, generating confidence intervals for evaluation metrics when comparing different models can be helpful in preventing erroneous conclusions from being drawn.

This is particularly important when working with small datasets, as certain regions of the feature space may be underrepresented. In such cases, it is better to predict with a margin of error, rather than point estimates. Although models on small data will have large confidence intervals, being aware of the range of predictions can be beneficial when making actionable decisions. By taking into account the uncertainty present in the problem, we can make more informed decisions about the trustworthiness of the model's predictions.

There are two other important and relevant (tangential) issues with deep learning. First, the model might be overconfident even if it is gravely wrong. Second, the model may still try to assign a confidence score to out-of-distribution samples. For

example, if a user feeds an image of a healthy chest X-ray to a Covid classifier model, the model should return a prediction with a high level of confidence. However, when a user uploads an image of head CT and asks the model to predict the Covid status, the model is faced with a situation of out-of-distribution test data. The model has been trained on images of X-rays, but has never seen a head CT before. The image of the head CT lies outside of the distribution of data the model was trained on.

This example can be extended other settings, such as MRI scans with structures that a diagnostic system has never observed before, or scenes that an autonomous car steering system has never been trained on. In such cases, a desirable behavior for the model would be to return a prediction, but also to convey that the point lies outside of the data distribution and therefore, the model possesses some quantity conveying a high level of uncertainty with such inputs [52].

Understanding the model uncertainty also plays a crucial role in an active learning framework. By recognizing which unlabelled data points would be the most beneficial to learn from, the model can make informed decisions on which data points to request labels for from an external resource such as a human annotator. These data points are selected using an acquisition function, which evaluates the potential value of each point for learning. Various acquisition functions exist, many of which factor in the model's uncertainty about the unlabelled data to make these decisions. By utilizing this approach, it allows for a more efficient use of limited labelled data, resulting in a more robust model [127].

3.3.20 Active learning

Active learning is a semi-supervised approach that can be leveraged when large amounts of data are present but obtaining labeled data is costly. Instead of randomly labeling data, active learning allows teams to strategically select the data points that will have the greatest impact on the performance of the model. In other words, the key idea behind active learning is that a ML algorithm can achieve greater accuracy with fewer training labels if it is allowed to choose the data from which it learns (i.e. with a limited number of training labels by actively selecting data for learning). The process is outlined in the active learning loop

in Figure3.19. In each iteration, the ML model is trained with a growing dataset created by labeling new data selected from a pool of unlabeled data.

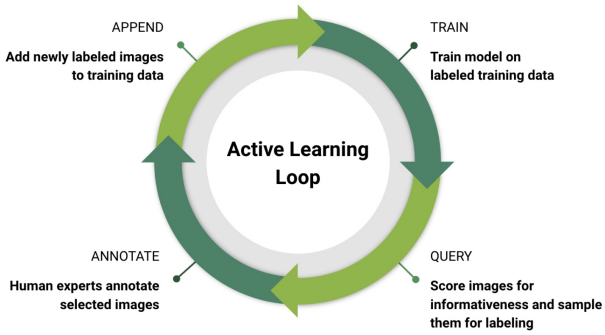


Figure 3.19: The Active learning loop diagram depicts the process of incrementally increasing the training dataset through selective labeling. In each iteration, the query step utilizes a scoring function and sampling strategy to determine which images should be added to the training dataset for further training after being labeled (source: [128]).

One way to implement active learning is by utilizing ML models for conducting preliminary tasks. These models can identify samples that are hard to classify, and then a human annotator can focus on labeling only those samples (selecting the next best data points to label). This way, the model can learn from the most informative data points, leading to more accurate predictions.

To further illustrate the concept of active learning, consider the analogy of a teacher and a student. Passive learning is like a student sitting in a class, listening to the teacher's lecture without engaging. On the other hand, active learning is when the student is an active participant, asking questions and collaborating with the teacher. The teacher in this scenario, focuses on the student's needs and spends more time on the concepts that are hard for the student to grasp. In this way, both the student and teacher are actively engaged in the learning process.

Similarly, in ML development, active learning is a collaborative process between the annotator and the modeler. The annotator provides a small labeled dataset, and the modeler uses it to train the model. The model then generates feedback on which data points should be labeled next. These models can identify samples that

are hard to classify, and then a human annotator can focus on labeling only those samples. This way, the model can learn from the most informative data points, leading to more accurate predictions. Through several iterations, the team can develop a more accurate model and a labeled gold training set.

Active learning is a training strategy that aims to reduce the amount of data that requires human labeling by intelligently (actively) selecting the most informative examples for labeling [129] where the learning algorithm is provided with a large pool of unlabeled data points, with the ability to request the labeling of any given examples from the unlabeled set in an interactive manner. This is accomplished through the use of a query strategy, which is applied to a ML model that has been initially trained on a small number of seed labeled examples. The process of labeling additional examples and updating the model is then repeated until the model is adequately trained. Active learning has been applied to a variety of ML tasks, including classification, detection, segmentation, and regression.

This approach is different from classical passive learning, where examples to be labeled are chosen randomly from the unlabeled pool. Instead, active learning aims to carefully choose the examples to be labeled in order to achieve a higher accuracy while using as few requests as possible, thereby minimizing the cost of obtaining labeled data. This approach is of particular interest in problems where data may be abundant, but labels are scarce or expensive to obtain. By carefully selecting the examples to be labeled, active learning allows to make the most of the available data while minimizing the cost of obtaining labeled data (keeping the annotation efforts at a minimum), making it useful in various real-world applications where labeled data is scarce and expensive [113].

In order to effectively identify difficult data points in active learning, a various combinations of methods are used to select the most informative samples for annotation (i.e. the process of identifying the most valuable examples to label next). These methods include classification uncertainty sampling, margin uncertainty, entropy sampling, disagreement-based sampling, information density, and business value [129]. An example of such a method can be found in the object detection problem. In this context, [128] proposed an image level scoring function that evaluates the usefulness of each unlabeled image for training. The selected images,

after being labeled, are then added to the training set. The experiments demonstrate the effectiveness of this approach, with up to 4 times relative mean average precision improvement compared to manual selection by experts.

Classification uncertainty sampling involves selecting the samples with the highest uncertainty, or the data points that the model knows the least about. By labeling these samples, the model becomes more knowledgeable and its performance improves [130].

Margin uncertainty, on the other hand, involves selecting the samples with the smallest margin. These are data points that the model knows about but is not confident enough to make good classifications. Labeling these examples increases the model's accuracy [131].

Diversity sampling aims to gather a representative sample of the entire data distribution, recognizing the significance of diversity as the model should perform well on any data encountered, not just a limited subset. The selected samples should reflect the underlying distribution. Common methods frequently depend on measuring the relationship between samples [132].

Entropy sampling uses entropy as a measure of uncertainty, which is proportional to the average number of guesses one has to make to find the true class. In this approach, we pick the samples with the highest entropy [130].

Disagreement-based sampling focuses on those samples where different algorithms disagree (asking annotators to label the examples on which classifiers disagree) [133]. For measuring the level of disagreement, there are several approaches used such as Kullback-Leibler (KL) divergence [134] and Jensen-Shannon divergence [135].

The **information density method** focuses on a denser region of data and select few points in each dense region. Labeling these data points help the model classify large number of data points around these points [136].

Lastly, the **business value method** focuses on labeling the data points that have higher business value than the others. This approach helps to prioritize labeling

the samples that are most important for the specific use-case and improve the model performance in real-world scenarios [137].

In active learning, the most informative examples are the ones that the classifier is the least certain about (a model has the least certainty are often the most challenging examples to classify). These examples are typically located near the class boundaries, and as a result, provide valuable information about the boundary between different classes.

In the field of ML, it is commonly understood that examples for which a model has the least certainty are often the most challenging examples to classify. These examples are typically located near the class boundaries, and as a result, provide valuable information about the boundary between different classes.

For instance, consider a binary classification problem in the medical domain where the task is to differentiate between benign and malignant tumors. The model may have a high level of certainty when presented with a clear and well-defined benign tumor, but may struggle to classify a malignant tumor that has similar features to a benign one. In this scenario, the model is uncertain about the correct classification of the example, as it lies near the class boundary between benign and malignant tumors. By observing this difficult example, the model can gain valuable information about the features that distinguish benign and malignant tumors and improve its performance on similar examples in the future.

Furthermore, research [129] has shown that models that are trained on a diverse set of examples, including difficult examples near class boundaries, tend to perform better on unseen data compared to models that are trained on a more homogeneous set of examples. Thus, it is crucial for the learning algorithm to identify and observe the difficult examples in order to gain a deeper understanding of the underlying data distribution and improve its overall performance.

In summary, the intuition behind the importance of difficult examples in ML is that they provide valuable information about the class boundaries and improve the performance of the model on unseen data. It is essential for the learning algorithm to identify and observe these difficult examples in order to gain a deeper understanding of the underlying data features.

Detailed overview of active learning concepts and methods can be found at [129].

3.3.21 Self-learning

Self-learning is a method that lies between semi-supervised and weakly supervised learning [138], utilizing an existing classifier to generate pseudo-labels for unlabeled data to train a new model. It operates by iteratively learning a classifier by assigning pseudo-labels to a subset of unlabeled training data with a margin greater than a set threshold. These pseudo-labeled samples are then combined with labeled training data to train a new classifier. The goal of pseudo-labeling is to generate labels for unlabeled data and improve the model's training with more information than before [139].

In contrast to self-supervised learning, where the model relies on the underlying structure of data to predict outcome (does not utilize labeled data), self-learning use both labeled and unlabeled data allowing the model (or models) to learn from themselves (or each other).

Another type of self-learning is called two-classifier self-training, which is similar to pseudo-labeling but utilizes two classifiers instead of one. The process involves training a classifier on the available data, then making predictions on the next batch of new data, alternating the classifier being used several times. This approach aims to improve the model's robustness by continuously feeding the output of one classifier as input to the other (i.e. each model learns on the output of the other) [140].

3.3.22 Multi-task learning

Multi-Task Learning (MTL) is a ML technique that leverages information from related tasks to improve performance on a primary task by training a model with multiple losses to perform well on multiple tasks (i.e., more than one loss function is optimized) [141].

In traditional ML, a single model or ensemble of models is trained to optimize for a specific metric, such as a benchmark score or a business KPI. However, by focusing solely on this single task, valuable information from related tasks is ignored. MTL

aims to overcome this limitation by sharing representations between related tasks, resulting in better generalization on the primary task.

For instance, in healthcare, a model trained to predict multiple medical conditions based on patient data can utilize information from related tasks, such as identifying risk factors, to improve predictions for each condition. Likewise, in agriculture, a model trained to predict crop yields for multiple crops can share representations between tasks, such as identifying soil characteristics, to improve predictions for each crop.

From a biological perspective, MTL can be considered modeled after human learning. In this context, when we learn new tasks, we tend to utilize the skills we have gained from related tasks. For instance, a baby starts by developing the ability to recognize faces and then applies that skill to identifying other objects.

MTL works effectively by incorporating two important concepts: Implicit data augmentation and Attention focusing [141].

Implicit data augmentation refers to the idea that by training a model on multiple tasks simultaneously, the model is effectively exposed to a larger sample size, leading to a better generalization of the model. This is because different tasks often have different patterns of noise, and training a model on multiple tasks helps average out these patterns and learn a more general representation.

Attention focusing is another advantage of multi-task learning. In limited and high-dimensional datasets or noisy tasks, it can be difficult for a model to identify relevant features. By training on multiple tasks, the model is able to focus its attention on important features as the additional tasks provide evidence for the relevance or irrelevance of these features.

When labeled data for the desired task is not present, an alternative approach in MTL is to utilize a task that is opposite of the goal. This can be accomplished through an adversarial loss that maximizes training error by using a gradient reversal layer. The adversarial task is predicting the input's domain, and by maximizing the adversarial task loss through gradient reversal, the model is forced to learn representations that are indifferent to the domains, which altogether

benefits the main task [141].

In summary, MTL provides a way for a model to learn more general representations and focus on important features that might not be easy to learn just using the original task, hence leading to improved performance on multiple tasks.

3.3.23 Symbolic learning

Symbolic learning is a form of ML that leverages symbolic representations of knowledge to make predictions with limited training data. By incorporating human-curated information into the learning process, this approach can effectively minimize the amount of training data required while enhancing the reliability and robustness of the ML system. This integration of human knowledge and ML capability also enables the creation of explainable ML systems, providing an opportunity to leverage a wealth of human knowledge to achieve performance outcomes that were not previously possible. Furthermore, this type of ML enhances the interaction between humans and ML systems by making ML's decisions more understandable to humans [142].

Symbolic models are fusing a representation of contextual knowledge into ML algorithms to improve algorithm performance [143]. These models utilize a representation of knowledge, often human-curated, such as an ontology, database, or contextual information. Traditional ML algorithms can be viewed as learning a representation of the features in the classes; in contrast, the symbolic model approach aims to reduce the amount of data required by introducing a human-constructed representation (knowledge).

Here are some examples:

- **Diagnosing diseases:** Imagine a patient presents with symptoms such as a cough, fever, and fatigue. A doctor might use his/her knowledge of common diseases to diagnose the patient with a respiratory infection. Similarly, symbolic learning algorithms can use information from medical ontologies or databases to diagnose diseases based on symptoms and other contextual information.

- **Predicting drug interactions:** When a patient is prescribed multiple medications, it is important to ensure that they do not interact in harmful ways. A pharmacist might use their knowledge of common drug interactions to determine the safety of a particular combination of drugs. Symbolic learning algorithms can be trained on data from drug databases to predict potential drug interactions and suggest alternatives if necessary.

- **Identifying risk factors for chronic diseases** Chronic diseases such as diabetes and heart disease are often the result of multiple risk factors, such as obesity, lack of exercise, and smoking. A doctor might use their knowledge of these risk factors to predict a patient's likelihood of developing a chronic disease. Symbolic learning algorithms can be trained on data from medical studies to identify and rank the most important risk factors for various chronic diseases.

- **Detecting plant diseases:** Diseases can significantly reduce crop yields and have a major impact on agriculture. Symbolic learning algorithms can be trained on data from plant disease databases to detect the presence of specific diseases based on symptoms such as discoloration of leaves or wilting of stems.

In each of these examples, symbolic learning is used to combine the strengths of ML algorithms and human knowledge to improve the accuracy and efficiency of decisions.

The selection of training samples is essential to any symbolic modelling efforts and not all data samples are equally informative: some carry unique information about the system, while others are redundant. To that end, [144] proposed an approach for constructing compact training data sets that serve as an input to a model learning method. For model learning, symbolic regression is chosen due to its ability to construct accurate models in the form of analytic equations even from small data sets as it, symbolic regression, outperforms other models for small data sets [145].

In summary, symbolic learning leverages human-curated information to perform limited data learning tasks and reduce the amount of data required for training. It

is a way of combining the strengths of traditional machine learning algorithms and human knowledge to improve the accuracy of predictions.

3.3.24 Hierarchical learning

Hierarchical learning is a ML approach that takes advantage of the hierarchical structure of real-world categories and taxonomies (i.e. using a taxonomy representing the relationship between objects and higher level classes) [146]. The hierarchical learning concept involves building a series of classification models that are structured based on a predefined hierarchy. The method utilizes transfer learning to build subsequent models, where the output of one model serves as input for the following. For instance, in healthcare, we can use hierarchical learning to diagnose diseases based on symptoms. A patient with a headache and nausea could be diagnosed with a migraine, but the model can also express uncertainty about the specific type of migraine (e.g., menstrual migraine or vestibular migraine). The hierarchical structure helps to make more accurate diagnoses based on limited data. In agriculture, hierarchical learning can be used to identify and classify different types of crops. For example, a model can classify a plant as a type of fruit, and then further classify it as an apple or a pear based on the shape, color, and other attributes of the fruit. This hierarchical structure helps to manage uncertainty in the classification process and makes it possible to train classifiers even when there is limited data available for certain types of crops.

[147] proposed a two-step framework using hierarchy transfer learning to build deep learning models for disease detection and classification, which achieved high accuracy and improved performance compared to other training approaches. The framework was extended to handle multiple input images and improved accuracy was achieved using a stacking ensembled method, leading to an improved performance even with a limited number of images.

3.3.25 Knowledge distillation based learning

Knowledge Distillation is a technique utilized to transfer the knowledge acquired by a complex model to a simpler model, with the goal of achieving similar or better performance on unseen data [148]. One of the key advantages of Knowledge Distillation is its ability to compress the knowledge of an ensemble of large base

models into a single, simpler model. This is particularly useful in situations where the storage and computational resources required by the ensemble of models is prohibitive. By distilling the knowledge of the complex model into a smaller, more tailored model, the efficiency and effectiveness of the model can be improved.

For example, when using a model trained on a dataset such as ImageNet, which contains 14 billion images and 100 classes, as a binary classifier for cats and dogs in a specific application, the computational resources required would be excessive. However, by using this model as a teacher and distilling its knowledge into a simpler model specifically designed for this task, the efficiency and effectiveness can be greatly improved.

To effectively transfer knowledge, it is important to examine the data used to train the network as it focuses on a specific area instead of the entire input space. However, access to this data may be restricted due to privacy concerns in various industries such as medicine, military, and industrial. In order to address this issue, [149] suggested KEGNET (Knowledge Extraction with Generative Networks), a new approach for knowledge distillation that does not require access to the original data. KEGNET learns the relationship between data points through training generator and decoder networks and generates artificial data to estimate the missing data on the manifold. At the same time, [150] argues that Knowledge Distillation is not as effective as widely believed, as there is often a significant gap between the teacher and student's predictive distributions, even when the student has the ability to exactly imitate the teacher.

3.4 Dealing with imbalanced data

In the scientific literature, data imbalance refers to the situation in which the number of observations in one class significantly exceeds those in other classes. This issue is often encountered in tasks such as detecting anomalies in electricity usage or identifying rare diseases, where the minority class (anomalies or rare cases) is of particular interest. If conventional ML algorithms are applied to imbalanced data without addressing this issue, the resulting models may be biased and inaccurate, as these algorithms are typically designed to minimize error assuming the class distribution is well-representing and unbiased. Examples of business problems

with imbalanced datasets include detecting rare diseases in medical diagnostics, identifying customer churn in the telecommunications industry, and predicting natural disasters such as earthquakes.

The problem of unbalanced data, in which the frequency of minority class observations is significantly lower than that of the majority class, can be addressed by balancing the data through sampling techniques. This can be achieved through increasing the frequency of the minority class (oversampling) or decreasing the frequency of the majority class (undersampling). The choice of oversampling versus undersampling and random versus clustered sampling depends on the size of the overall dataset and the distribution of the data. In general, oversampling is preferred when the dataset is small, while undersampling is more suitable when the dataset is large. Similarly, the decision between random and clustered sampling is influenced by the distribution of the data.

Oversampling (up-sampling): One approach to addressing imbalanced data is over-sampling, which involves increasing the number of instances in the minority class by randomly replicating them in order to present a higher representation of the minority class in the sample. This method has the advantage of not leading to information loss, and it has been shown to outperform under-sampling. However, it also has the disadvantage of increasing the likelihood of overfitting due to the replication of minority class events.

Another up-sampling technique is K-means clustering that is independently applied to minority and majority class instances. This approach involves identifying clusters in the dataset and oversampling each cluster such that all clusters of the same class have an equal number of instances and all classes have the same size. While this technique has the advantage of addressing both inter- and intra-class imbalances, it also carries the risk of overfitting the training data. Overall, K-means clustering is a useful method for addressing imbalanced data, but it should be used with caution to avoid compromising the generalizability of the model.

Undersampling (down-sampling): In the context of addressing imbalanced data, undersampling involves randomly eliminating majority class examples in order to balance the class distribution. While this technique can be useful for improving run time and storage issues when dealing with large training datasets,

it is also important to consider its potential drawbacks, such as the potential for discarding potentially useful information and the risk of introducing bias through the random selection of samples. Additionally, the resulting sample may not accurately represent the overall population, leading to potentially inaccurate results when applied to the actual test dataset.

There are other techniques to allow the model learn under extreme imbalanced datasets. A concrete technique that adds a final batch normalization layer is one of this work's contributions and will be explored in detail in Chapter 4.

3.5 Anomaly Detection as a Small Data Problem

Anomaly detection, also known as outlier detection or novelty detection, is the task of identifying unusual or abnormal data points in a dataset. It is a crucial task in a wide range of applications, including fraud detection, cybersecurity, and quality control.

Detecting anomalies that are hardly distinguishable from the majority of observations is a challenging task that often requires strong learning capabilities. Since anomalies appear scarcely, and in instances of diverse nature, a labeled dataset representative of all forms is typically unattainable. In some cases, such as working with underrepresented populations or studying rare medical conditions, only limited data are available [61].

Despite tremendous advances in computer vision and object recognition algorithms, their effectiveness remains strongly dependent upon the size and distribution of the training set. Real-world settings dictate hard limitations on training sets of rarely recorded events in Agriculture or Healthcare, e.g., early stages of certain crop diseases or premature malignant tumors in humans.

Anomaly detection is mostly concerned with hard classification problems at early-stages of abnormalities in certain domains (i.e. crop, human diseases, chip manufacturing), which suffer from lack of data instances, and whose effective treatment would make a dramatic impact in these domains. For instance, fungus' visual cues on crops in agriculture or early-stage malignant tumors in the medical domain are hardly detectable in the relevant time-window, while the highly infectious nature

leads rapidly to devastation in a large scale. Other examples include detecting the faults in chip manufacturing industry, automated insulation defect detection with thermography data, assessments of installed solar capacity based on earth observation data, and nature reserve monitoring with remote sensing and deep learning. However, class imbalance poses an obstacle when addressing each of these applications.

In Precision Agriculture, and particularly in Precision Crop Protection [151], certain visual cues must be recognized with high accuracy in early stages of infectious diseases' development. A renowned use-case is the Potato Late Blight, with dramatic historical and economical impacts [152], whose early detection in field settings remains an open challenge (despite progress achieved in related learning tasks; see, e.g., [153]). The hard challenge stems from the actual nature of the visual cues (which resemble soil stains and are hardly distinguishable), but primarily from the fact that well-recording those early-stage indications is a rare event.

In the case of highly imbalanced datasets, such as those involving fraud or machine failure, it may be worth considering whether these examples can be classified as anomalies. If the problem meets the criteria for anomaly detection, techniques such as OneClassSVM, clustering methods, or Gaussian anomaly detection methods may be employed. These approaches involve considering the minority class as the outlier class, potentially providing new ways to classify and differentiate. Change detection is a similar concept to anomaly detection, but focuses on identifying changes or differences rather than anomalies. Examples of this include changes in user behavior as indicated by usage patterns or bank transactions.

In recent years, reliable models capable of learning from small samples have been obtained through various approaches, such as autoencoders [154], class-balanced loss (CBL) to find the effective number of samples required [155], fine tuning with transfer learning [156], data augmentation [75], cosine loss utilizing (replacing categorical cross entropy) [66], or prior knowledge [48].

3.6 Conclusion

As we conclude Chapter 3, our exploration through the diverse landscape of strategies and methodologies essential for effective learning from small datasets has been both comprehensive and insightful. This journey has armed us with a robust toolkit, encompassing a broad spectrum of techniques ranging from data augmentation, ensemble methods, and transfer learning to more intricate strategies like parameter initialization, loss function reformulation, and regularization techniques. We have also ventured into advanced realms such as synthetic data generation, physics-informed neural networks, and various forms of learning including unsupervised, semi-supervised, self-supervised, zero-shot, one-shot, few-shot, metalearning, and beyond. These techniques, together with insights into tackling specific challenges like imbalanced data and anomaly detection, form a crucial foundation for the focused applications and advanced methods explored in the subsequent chapters.

In Chapter 4 and 5, we build directly upon this foundation, delving into the nuances of learning from imbalanced datasets and exploiting the potential of self-supervised learning in scenarios of limited data. We have laid the groundwork for understanding how each of these advanced techniques and strategies can be effectively applied to real-world problems where data scarcity is a significant challenge. This chapter, therefore, acts as a pivotal link in our dissertation, ensuring a seamless and cohesive narrative that underscores the importance of mastering small data learning. Ultimately, the insights and methodologies discussed here contribute to the development of more robust and efficient machine learning models, capable of addressing the complex challenges presented by small datasets in various domains.

As we draw Chapter 3 to a close, we reflect on the extensive exploration of foundational concepts and methodologies critical for learning effectively from small datasets. The techniques and insights garnered here, from data augmentation to transfer learning and beyond, are not just theoretical constructs but essential tools that underpin the advanced topics in Chapters 4 and 5. The upcoming chapters build upon this bedrock, delving into the nuances of learning from imbalanced datasets (Chapter 4) and exploring the potential of self-supervised learning with salient image segmentation (Chapter 5), both of which hinge on the principles

established in this chapter. By connecting these concepts, we ensure a seamless transition into the intricacies of batch normalization and the dynamics of image segmentation in self-supervised learning. This chapter, thus, is a bridge that not only links but also enriches the narrative flow of the dissertation, ensuring that the journey from foundational principles to advanced applications in machine learning is both cohesive and comprehensive.

Chapter 4

The Role of Final Batch Normalization Layer

In highly imbalanced classification problems, which encompass complex features, deep learning (DL) is much needed because of its strong detection capabilities. At the same time, DL is observed in practice to favor majority over minority classes and consequently suffer from inaccurate detection of rare events. To simulate this scenario, in this chapter, we will artificially generate skewness (99% vs. 1%) for certain plant types out of the PlantVillage dataset [157] as a basis for classification of scarce visual cues through transfer learning. By randomly and unevenly picking healthy and unhealthy samples from certain plant types to form a training set, we consider a base experiment as fine-tuning ResNet34 and VGG19 architectures and then testing the model performance on a balanced dataset of healthy and unhealthy images. Then we will investigate the role of Batch Normalization (BN) layer in modern CNN architectures to see if it would help minimize the training time and testing error for minority classes in highly imbalanced data sets.

In this chapter we shall present the following results:

1. Utilizing an additional Batch Normalization (BN) layer before the output

layer in modern CNN architectures has a considerable impact in terms of minimizing the training time and testing error for minority classes in highly imbalanced data sets.

2. When the final BN is employed, minimizing the loss function may not be the best way to assure a high F1 test score for minority classes in such problems. That is, the network might perform better even if it is not 'confident' enough while making a prediction; leading to another discussion about why softmax output is not a good uncertainty measure for DL models.

3. The performance gain after adding the final BN layer in highly imbalanced settings could still be achieved after removing this additional BN layer in inference.

4. There is a certain threshold for the *imbalance ratio* upon which the progress gained by the final BN layer reaches its peak.

5. The batch size also plays a role and affects the outcome of the final BN application.

6. The impact of the BN application is also reproducible on other datasets and when utilizing much simpler neural architectures.

7. The reported BN effect occurs only per a single majority class and multiple minority classes – i.e., no improvements are evident when there are two majority classes; and finally, (viii) Utilizing this BN layer with sigmoid activation has almost no impact when dealing with a strongly imbalanced image classification tasks.

4.1 Background

In order to better understand the novel contributions of this study, in this section, we give some background information about the Batch Normalization [74] concept.

Training deep neural networks with dozens of layers is challenging as they can be sensitive to the initial random weights and configuration of the learning algorithm. One possible reason for this difficulty is that the distribution of the inputs to

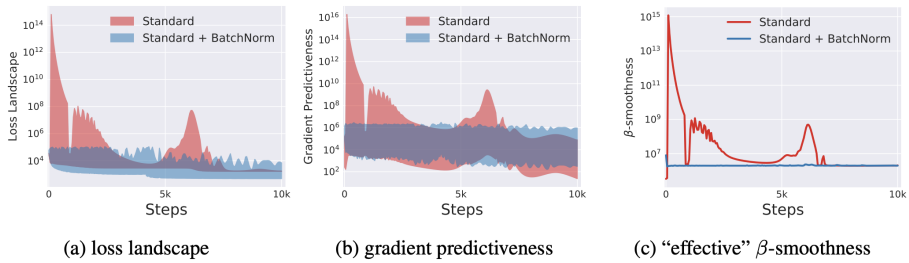
layers deep in the network may change after each mini-batch when the weights are updated. This slows down the training by requiring lower learning rates and careful parameter initialization, makes it notoriously hard to train models with saturating nonlinearities [74], and can cause the learning algorithm to forever chase a moving target. This change in the distribution of inputs to layers in the network is referred to by the technical name "internal covariate shift" (ICS).

BN is a widely adopted technique that is designed to combat ICS and to enable faster and more stable training of deep neural networks (DNNs). It is an operation added to the model before activation which normalizes the inputs and then applies learnable scale (γ) and shift (β) parameters to preserve model performance. Given m activation values $x_1 \ldots, x_m$ from a mini-batch $\mathcal{B}$ for any particular layer input $x^{(j)}$ and any dimension $j \in \{1, \ldots, d\}$, the transformation uses the mini-batch mean $\mu_{\mathcal{B}} = 1/m \sum_{i=1}^{m} x_i$ and variance $\sigma_{\mathcal{B}}^2 = 1/m \sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2$ for normalizing the x_i according to $\hat{x}_i = (x_i - \mu_{\mathcal{B}})/\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}$ and then applies the scale and shift to obtain the transformed values $y_i = \gamma \hat{x}_i + \beta$. The constant $\epsilon > 0$ assures numerical stability of the transformation.

Typically in ML, it is common to normalize input data before passing the data to the input layer. The reason we normalize is partly to ensure that our model can generalize appropriately. This is achieved by ensuring that the scale of the values is balanced, and also the range of the values are maintained and proportional despite the scale change in the values. In a similar context, the BN operation standardizes and normalizes the input values. The input values are then transformed through scaling and shifting operations. BN was performed as a solution to speed up the training phase of deep neural networks through the introduction of internal normalization of the inputs values within the neural network layer. Normalization is typically carried out on the input data, but it would make sense that the flow of internal data within the network should remain normalized. So, we can say that BN is the internal enforcer of normalization within the input values passed between the layer of a neural network. Internal normalization limits the covariate shift that usually occurs to the activations within the layers.

BN has the effect of stabilizing the learning process and dramatically reducing the number of training epochs required to train deep networks; and using BN

makes the network more stable during training. This may require the use of much larger than normal learning rates, which in turn may further speed up the learning process. Though BN has been around for a few years and has become common in deep architectures, it remains one of the DL concepts that is not fully understood, having many studies discussing why and how it works. Most notably, Santurkar et al. [158] recently demonstrated that such distributional stability of layer inputs has little to do with the success of BN and the relationship between ICS and BN is tenuous. Instead, they uncovered a more fundamental impact of BN on the training process: it makes the optimization landscape significantly smoother (Figure 4.1). This smoothness induces a more predictive and stable behavior of the gradients, allowing for faster training. Bjorck et al. [159] also makes similar statements that the success of BN can be explained without ICS. They argue that being able to use larger learning rate increases the implicit regularization of the gradient, which improves generalization.



(a) loss landscape (b) gradient predictiveness (c) "effective" β-smoothness

Figure 4.1: Analysis of the optimization landscape during training of deep linear networks with and without BatchNorm. There is a clear improvement in each of these measures of smoothness of the optimization landscape in networks with BatchNorm layers. (source: [158])

Even though BN adds an overhead to each iteration (estimated as additional 30% computation [160]), the following advantages of BN outweigh the overhead shortcoming:

- It improves gradient flow and allows training deeper models (e.g., ResNet).
- It enables using higher learning rates because it eliminates outliers activation, hence the learning process may be accelerated using those high rates.

- It reduces the dependency on initialization and then reduces overfitting due to its minor regularization effect. Similarly to dropout, it adds some noise to each hidden layer's activation.

- Since the scale of input features would not differ significantly, the gradient descent may reduce the oscillations when approaching the optimum and thus converge faster.

- BN reduces the impacts of earlier layers on the following layers in DNNs. Therefore, it takes more time to train the model to converge. However, the use of BN can reduce the impact of earlier layers by keeping the mean and variance fixed, which in some way makes the layers independent from each other. Consequently, the convergence becomes faster.

The development of BN as a normalization technique was a turning point in the development of DL models, and it enabled various networks to train and converge. Despite its great success, BN exhibits drawbacks that are caused by its distinct behavior of normalizing along the batch dimension. One of the major disadvantages of BN is that it requires sufficiently large batch sizes to obtain good results. This prevents the user from exploring higher-capacity models that would be limited by memory. To solve this problem, several other normalization variants are developed, such as Layer Normalization (LN) [161], Instance Normalization (IN) [162], Group Normalization (GN) [163] and Filter Response Normalization (FRN) [164].

4.2 Related Work and Prior Art

Investigating the effect of learnable parameters of BN, scale (γ) and shift (β), on the training of various typical deep neural nets, Wang et al. [165] suggest that there is no big difference in both training convergence and final test accuracy when removing the BN layer following the final convolutional layer from a convolutional neural network (CNN) for standard classification tasks. The authors claim that it is not necessary to adjust the learnable gain and bias along the training process and they show that the use of constant gain and bias is enough and these learneable parameters have little effect on the performance. They also observe that without adaptively updating learnable parameters for BN layers, it often requires less time

for training of very deep neural nets such as ResNet-101.

Frankle et al. [166] also studied the effect of learnable BN parameters, showing that BN's trainable parameters alone can account for much of a network's accuracy. They found that, when locking all other layers at their random initial weights and then training the network for fifty or so epochs, it will perform better than random. As it adjusts a network's intermediate feature representations for a given mini batch, BN itself learns how to do so in a consistent way for all mini batches. The researchers probed the impact of this learning by training only the BN parameters, γ and β, while setting all other parameters at random. Bjorck et al. [159] conducted another experiment, similar to our study. They trained a ResNet that uses one BN layer only at the very last convolutional layer of the network (removing all the other BN layers coming after each CNN layer), normalizing the output of the last residual block but without intermediate activation. This indicates that after initialization, the network tends to almost always predict the same (typically wrong) class, which is then corrected with a strong gradient update. In contrast, the network with BN does not exhibit the same behavior; rather, positive gradients are distributed throughout all classes. Their findings suggest that normalizing the final layer of a deep network may be one of the most important contributions of BN.

Zhu et al. [167] also adopted a similar idea of using the scalable version of a BN layer to normalize the channel at the final output of the network, and improving the classification performance of softmax without adding learnable BN parameters, but controlling the output distribution by another scaling parameter. However, their improvements on selected datasets were not significant (less then 0.3%).

Learning from small samples can also be regarded as out-of-distribution (OoD) detection as the minority class represents the samples out of the distribution of majority classes. While investigating the OoD image detection capability of neural networks without Learning from OoD data, [168] proposed Generalised ODIN (i.e. OoD image detection in neural networks [169]) framework that is composed of two strategies: decomposing confidence scoring and modifying the input pre-processing method. The original ODIN framework [169] also suggests that temperature scaling and adding small perturbations to the input can separate

the softmax score distributions of in- and out-of-distribution images, allowing for more effective detection. Both of these frameworks tamper the confidence scoring in the output layer but mentions BN layer within a regularization context.

Apart from these studies, to the best of our knowledge, there is no prior work investigating the effect of BN utilization over imbalanced classification tasks when set before the softmax output layer.

4.3 Implementation Details and Experimental Results

In this study, we primarily utilize the ResNet34 CNN architecture due to computational requirements and a need for an iterative process to conduct a large number of experiments. ResNet is evaluated on the ImageNet with a depth of up to 152 layers - 8 times deeper than VGG nets but still having lower complexity. An ensemble of these residual nets achieved a 3.57% error on the ImageNet test set. This result won the 1st place in the ILSVRC 2015 classification task [170].

Our training process can be regarded as fine-tuning based on ImageNet checkpoints using transfer learning. We firstly addressed the complete PlantVillage original dataset and fine-tuned a ResNet34 model for 38 classes. Using scheduled learning rates, we obtained 99.782% accuracy after 10 epochs – slightly improving the PlantVillage project's record of 99.34% using GoogleNet [171]. Having reproducing these results, relatively easily, further boosts our confidence in selecting ResNet34 for the task.

4.3.1 Dataset Details

A prominent effort towards plant diseases' detection is the Plant Village Disease Classification Challenge. As part of the PlantVillage project, 54,306 images of 14 crop species with 26 diseases (including healthy, 38 classes in total) were made publicly available [157]. The detailed distribution of the dataset and its classes can be seen at Table 4.1.

Following the release of the PlantVillage dataset, deep learning (DL) was intensively employed [171], with reported accuracy ranging from 85.53% to 99.34%. Without

Table 4.1: Plant Village dataset distribution for 54,306 images of 14 crop species with 26 diseases. Each specie has certain number of *unhealthy* classes (e.g. apple scab and black rot for Apple, bacterial spot and leaf mold for Tomato) and one *healthy* class, denoted in Class Size column as the total number of classes.

	Training Set		Validation Set		Class Size
	Healthy	Unhealthy	Healthy	Unhealthy	
Potato	121	1600	31	400	3
Peach	288	1838	72	459	2
Cherry	684	842	170	210	2
Grape	339	2912	84	727	4
Tomato	1272	13132	318	3284	10
Pepper	1181	797	295	200	2
Corn	16	2005	5		4
Orange	0	4405	0	1102	2
Blueberry	1202	0	300	0	2
Apple	1316	1220	329	306	4
Squash	0	1448	0	365	2
Soybean	4072	0	1018	0	2
Raspberry	297	0	74	0	2
Strawberry	364	886	92	222	2

any feature engineering, the best reported model correctly classifies crop and disease in 993 out of 1,000 images (out of 38 possible classes). The PlantVillage dataset is slightly imbalanced, such that accuracy is commonly used as the performance measure.

In this study, to simulate rare events in agriculture, we capitalize on this dataset to synthetically generate skewness and address our research question. With the first model, using ResNet34 architecture [170] with pre-trained weights on ImageNet and tuning the model for PlantVillage data sets, we reach 99.782% accuracy after 10 epochs, i.e., we were able to correctly classify crop and disease from 38 possible classes in 998 out of 1,000 images. This result can be attributed to the representation power of the ResNet architecture as well as to having a large number of crop disease samples from different classes. Upon firstly reproducing the accuracy rates reported in [171] on the original dataset, we considered the classification problem under the generated imbalance.

We randomly picked 1,000 healthy and 10 unhealthy samples from a plant type for the training set, 150 healthy and 7 unhealthy samples for the validation set, fine tuned using the ResNet34 architecture, and then tested the model performance on an equal number of healthy and unhealthy images (150 vs 150) in a test set. We did this for 3 different plant types: Apple, Pepper, and Tomato. Training on just 10 samples and validating on 7 samples from minority class, we managed to correctly predict more than 141 of 150 unhealthy images spanning over multiple anomalies that may not exist in the training or validation set. When experimenting with different configurations, we discovered a significant improvement of classification performance by adding a final BN layer just before the output layer. We did more experiments in the VGG19 framework [172] under the same settings, and managed to correctly predict more than 143 of 150 unhealthy images, which is higher than what we got through ResNet34: 130 out of 150.

4.3.2 Adding a Final Batch Norm Layer Before the Output Layer

By using the imbalanced datasets for certain plant types (1,000 healthy/10 unhealthy samples in the training set, 150/7 in the validation set and 150/150 in

the test set), we performed several experiments with the VGG19 and ResNet34 architectures. The selected plant types were Apple, Pepper and Tomato - being the only datasets of sufficient size to enable the 99%-1% skewness generation. We treated unhealthy class as a minority class due to the fact that the abnormalities are rare in the real-world as well and gathering the unhealthy images is harder compared to healthy ones. So, we set the ratio of unhealthy images to 1% in train set and 5% in validation set. But we created a balanced dataset for test set to test the model performance on unseen images that are in an equal number for both classes.

In order to fine-tune our network for the PlantVillage dataset, the final classification layer of CNN architectures is replaced by Adaptive Average Pooling (AAP), BN, Dropout, Dense, ReLU, BN and Dropout followed by the Dense and BN layer again. The last layer of an image classification network is often a fully-connected layer with a hidden size being equal to the number of labels to output the predicted confidence scores that are normalized by the softmax operator to obtain predicted probabilities. In our implementation, we add another 2-input BN layer after the last dense layer (before softmax output) in addition to existing BN layers in the tail and 4 BN layers in the head of the DL architecture (e.g., ResNet34 possesses a BN layer after each convolutional layer, having altogether 38 BN layers given the additional 4 in the head). A schematic outline of this architecture is provided in Figure 4.2.

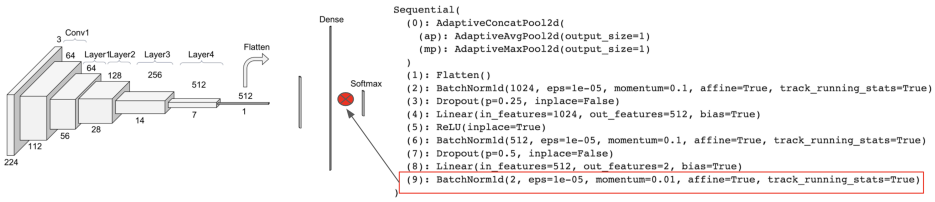


Figure 4.2: Implementation of the final BN layer in ResNet34, just before the softmax output.

At first we run experiments with VGG19 architectures for selected plant types by adding the final BN layer. When we train this model for 10 epochs and repeat this for 10 times, we observed that the F1 test score is increased from 0.2942 to 0.9562 for unhealthy Apple, from 0.7237 to 0.9575 for unhealthy Pepper and from 0.5688

to 0.9786 for unhealthy Tomato leaves. We also achieved substantial improvement in healthy samples (being the majority in the training set). For detailed metrics and charts, see Figure 4.3 and Table 4.2.

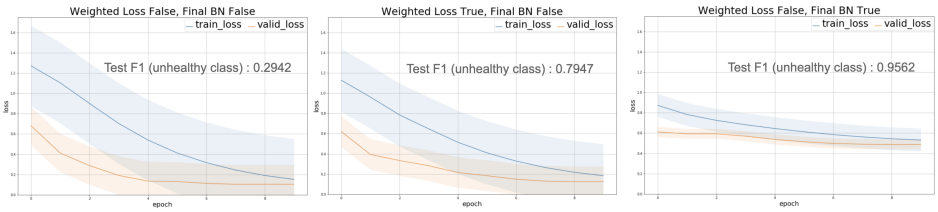


Figure 4.3: Averaged loss charts over 10 runs for the Apple dataset without final BN (left), with final BN layer (right) and with WL (center) in VGG19 architecture. Without WL, after adding the final BN layer, the test F1 score is increased from 0.2942 to 0.9562 for the minority class while the train and validation losses level off around 0.5 before the model begins to overfit. It is also evident that the standard deviation among runs (depicted as the shaded areas surrounding a given curve within each plot) is lower for the latter case, as the training becomes smoother therein.

Table 4.2: Averaged **F1 test set** performance values over 10 runs, alongside BN's total improvement, using 10 epochs with VGG19, with/without BN and with Weighted Loss (WL) without BN.

Plant	Class	without final BN	with WL (no BN)	with final BN (no WL)	BN total improvement
Apple	Unhealthy	**0.2942**	**0.7947**	**0.9562**	**0.1615**
	Healthy	0.7075	0.8596	0.9577	0.0981
Pepper	Unhealthy	**0.7237**	**0.8939**	**0.9575**	**0.0636**
	Healthy	0.8229	0.9121	0.9558	0.0437
Tomato	Unhealthy	**0.5688**	**0.8671**	**0.9786**	**0.1115**
	Healthy	0.7708	0.9121	0.9780	0.0659

In order to explain which parts of the image the network was looking at when it made a prediction, we experimented with class-discriminative activation maps (a generic localizable deep representation). Class activation maps [173], commonly called CAMs, are class-discriminative saliency maps. While saliency maps give information on the most important parts of an image for a particular class, class-discriminative saliency maps help distinguish between classes. We run experiments for *Apple* class with and wothout final BN layer using ResNet34 and visualized the activation maps. As you can see at Figure 4.4, the model without final BN layer

looks at the irrelevant parts of the picture and predicting wrong while it is looking at the right parts when the final BN layer is added.

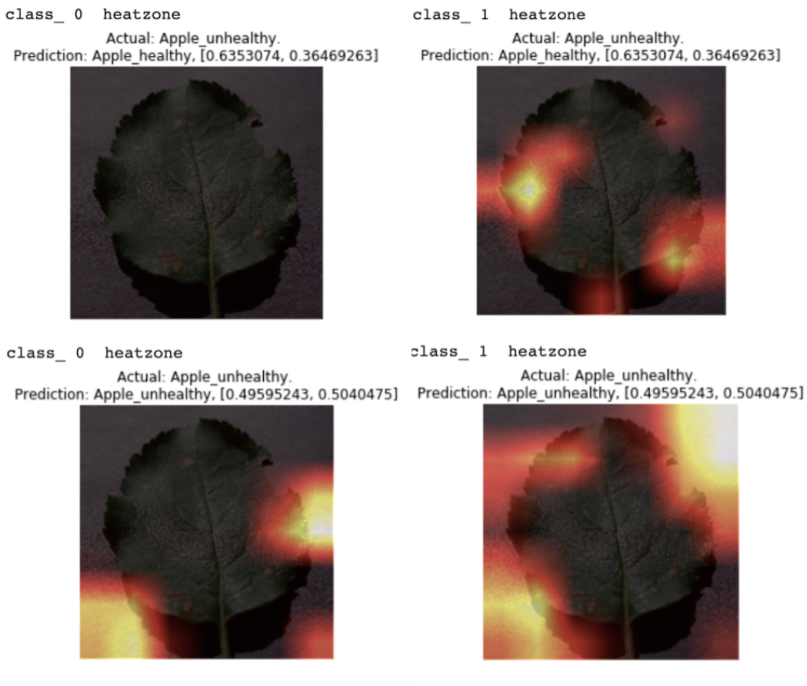


Figure 4.4: Class-discriminative activation maps of a sample prediction under two settings (with and without final BN layer). Class-0 indicates *Healthy* and Class-1 indicates *Unhealthy*. The first row belongs to the predictions under vanilla settings and the second row belongs to the predictions when final BN layer is added. With final BN, the activations for *Unhealthy* class is glowing over the wrong pixels and it is predicted as *Healthy*. The probability of being *Unhealthy* is 36.5% and we see some minor activations closer to blight areas while no activation is happening for *Healthy* class. On the other hand, with final BN, the activations for *Unhealthy* class is glowing over the right pixels and it is predicted as *Unhealthy*. The probability of being *Healthy* is 49.6% and we see some minor activations closer to *Healthy* areas.

The highlighted parts (heatmap) basically tells us which parts of an image induced the wrong classification. In this example, the model made the wrong prediction because of the highlighted part.

4.3.3 Experimentation Subject to Different Configurations

Using the following six configuration variations with two options each, we created 64 different configurations which we tested with ResNet34 (training for 10 epochs only): Adding (✓) a final BN layer just before the output layer (BN), using (✓) weighted cross-entropy loss [174] according to class imbalance (WL), using (✓) data augmentation (DA), using (✓) mixup (MX) [175], unfreezing (✓) or freezing (learnable vs pre-trained weights) the previous BN layers in ResNet34 (UF), and using (✓) weight decay (WD) [176]. Checkmarks (✓) and two-letter abbreviations are used in Table 4.3 and Table 4.4 to denote configurations. When an option is disabled across all configurations, its associated column is dropped.

Table 4.3: Best performance metrics over the Apple dataset under various configurations using ResNet34.

Class	Config Id	Test set precision	Test set recall	Test set F1-score	Epoch	BN	DA	UF	WD
Unhealthy	**31**	0.9856	0.9133	**0.9481**	6	✓			
(class = 1)	23	0.9718	0.9200	0.9452	6	✓	✓		
	20	0.9926	0.8933	0.9404	7	✓	✓	✓	✓
Healthy	31	0.9193	0.9867	0.9518	6	✓			
(class = 0)	23	0.9241	0.9733	0.9481	6	✓	✓		
	20	0.9030	0.9933	0.9460	7	✓	✓	✓	✓

As shown in Table 4.3, just adding the final BN layer was enough to get the highest F1 test score in both classes. Surprisingly, although there is already a BN layer after each convolutional layer in the base CNN architecture, adding one more BN layer just before the output layer boosts the test scores. Figure 4.5 shows that we gain a boost in Test F1 score by more than double (by 1.2 times) just by adding a final BN layer.

Notably, the 3rd best score (average score for configuration 31 in Table 4.4) is achieved just by adding a single BN layer before the output layer, even without unfreezing the previous BN layers. Moreover, it is evident that the additional BN layer is never utilized among the worst performing configurations (see Table 4.4).

The model without the final BN layer is pretty confident even if it predicts falsely. But the proposed model with the final BN layer predicts correctly even though

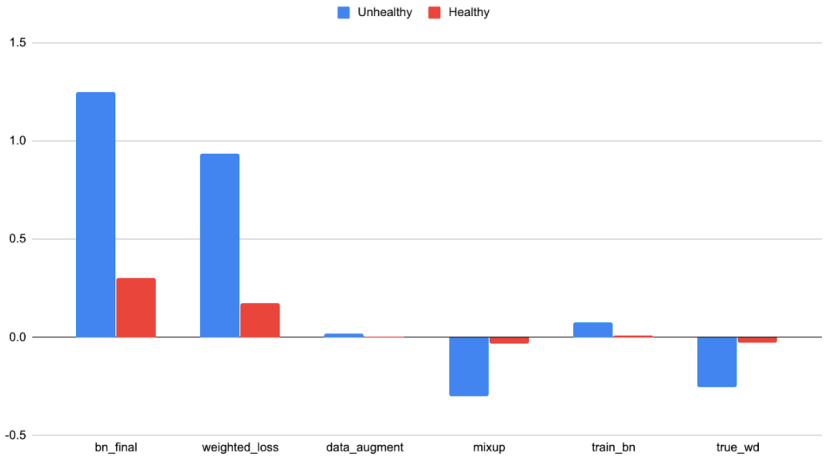


Figure 4.5: The ratio of increase in Test-F1 scores per each configuration. The highest relative gain in Test-F1 score for both classes is achieved by adding a final BN layer (only one parameter is set to True at a time). The experiments show that the enhancement is more than doubled just by adding a final BN layer (e.g., from 0.42 to 0.95 translates into a (0.95-0.42)/0.42 = 1.2 gain factor).

Table 4.4: Best (top three) and worst (bottom three) performing configurations (F1 measure, for the unhealthy/minority class) when using ResNet34 for the Apple, Pepper and Tomato datasets.

Config Id	Apple	Pepper	Tomato	Average	BN	DA	MX	UF	WD
29	0.9332	0.9700	0.9866	**0.9633**	✓			✓	
28	0.9332	0.9566	0.9966	0.9622	✓			✓	✓
31	0.9499	0.9433	0.9833	**0.9588**	✓				
49	0.6804	0.5080	0.5339	0.5741		✓	✓	✓	
48	0.5288	0.5638	0.5638	0.5521		✓	✓	✓	✓
50	0.6377	0.5027	0.4528	0.5311		✓	✓		✓

it is less confident. We end up with less confident but more accurate models in less than 10 epochs. The classification probabilities for five sample images from the unhealthy class (class = 1) with final BN layer (right column) and without final BN layer (left column) are shown in Table 4.5. As explained above, without the final BN layer, these anomalies are all falsely classified (recall that $\mathcal{P}_{\text{softmax}}(\text{class} = 0) = 1 - \mathcal{P}_{\text{softmax}}(\text{class} = 1)$).

Table 4.5: Softmax output values (representing class probabilities) for five sample images of unhealthy plants. Left column: Without final BN layer, softmax output values for unhealthy, resulting in a wrong classification in each case. Right column: With final BN layer, softmax output value for unhealthy, resulting in correct but less "confident" classifications.

Without final BN layer	With final BN layer
0.1082	0.5108
0.1464	0.6369
0.1999	0.6082
0.2725	0.6866
0.3338	0.7032

4.3.3.1 Removing the additional BN layer during inference

Since adding the final BN layer adds a small overhead (four new parameters) to the network at each iteration, we experimented if the final BN layer could be dropped once the training is finished so that we can avoid the cost. Dropping this final BN layer means that training the network from end to end, and then chopping off the final BN layer from the network before saving the weights. We tested this hypothesis for Apple, Pepper and Tomato images from PV dataset under three conditions with 1% imbalance ratio: *Without final BN, with final BN and then removing the final BN during testing.* We observed that removing the final BN layer in inference would still give us a considerable boost on minority class without losing any performance gain on the majority class. The results in Table 4.6 show that the performance gain is very close to the configuration in which we used the final BN layer both in training and inference time. As a consequence, we confirm an hypothesis that the final BN layer can indeed be removed in inference without compromising the performance gain.

Table 4.6: Training with the final BN layer, and then dropping this layer while evaluating on the test set proved to be still useful in terms of improving the classification score on minority classes, albeit not as much as with the final BN layer kept (imbalance ratio 0.01, epoch 10, batch size 64).

	Apple		Pepper		Tomoto	
	healthy	unhealthy	healthy	unhealthy	healthy	unhealthy
with no final BN	0.71	0.22	0.74	0.45	0.74	0.46
with final BN	0.92	0.91	0.94	0.94	0.98	0.98
train with final BN remove while testing	0.74	**0.83**	0.75	**0.82**	0.78	**0.85**

4.3.3.2 Impact level regarding the imbalance ratio and the batch size

In the previous sections, we empirically showed that the final BN layer, when placed before the softmax output layer, has a considerable impact in highly imbalanced image classification problems but they fail to explain the impact of using the final BN layer as a function of level of imbalance in the training set. In order to find if there is a certain ratio in which the impact is maximized, we tested this hypothesis (H-1) over various levels of imbalance ratios and conditions explained below. In sum, we ended up with 430 model runs, each with 10 epochs (basically tested with 5, 10, 15, .. 100 unhealthy vs 1000 healthy samples). During the experiments, we observed that the impact of final BN on highly imbalanced settings is the most obvious when the ratio of minority class to the majority is less than 10%; above that almost no impact. As expected, the impact of the final BN layer is more obvious on minority class than it is on majority class, albeit the level of impact with respect to the imbalance ratio is almost same, and levels off around 10%. It is mainly because of the fact that the backbone architecture (ResNet34) is already good enough to converge faster on such data set (Plant Village) and the model does well on both classes after 10% imbalance (having more than 100 unhealthy with respect to 1000 healthy samples can already be handled regardless of final BN trick). During these experiments, we also tested if unfreezing the previous layers in the backbone CNN architecture (ResNet34) would also matter. We observed that unfreezing the pretrained layers helps without even final BN layer, but unfreezing adds more computation as the gradient loss will be calculated for each one of them. After adding the final BN layer and freezing the previous pretrained layers, we observed similar metrics and learning pattern as we did with unfreezing but not

with final BN layer. This is another advantage of using the final BN that allow us to freeze the previous pretrained layers. The results are displayed as charts in Figure 4.6 and Figure 4.7.

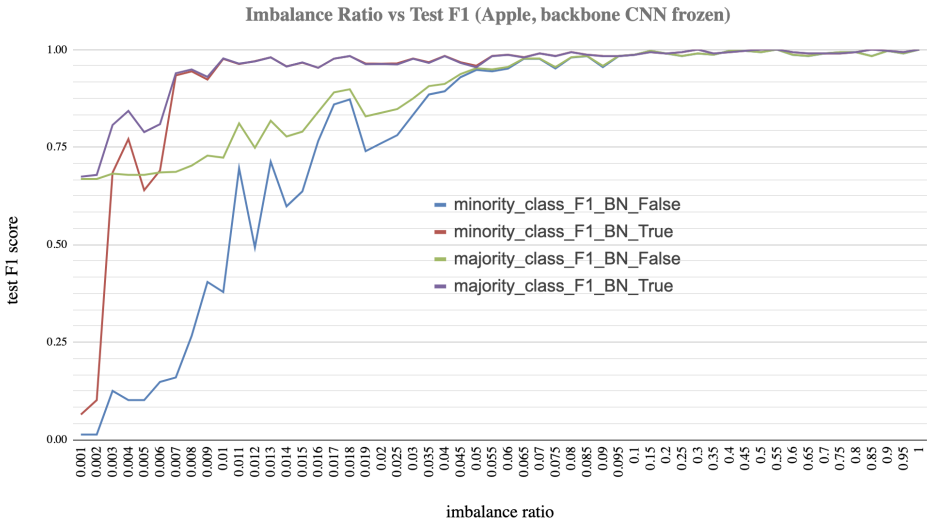


Figure 4.6: The impact of final BN layer on the F1 test score of each class for Apple plant. The impact is the most obvious when the ratio of minority class to the majority is less than 0.1.

We also experimented if the batch size would also be an important parameter for the minority class test accuracy when the final BN is added and found out that the highest score is achieved when the batch size is around 64, whereas the accuracy drops afterwards with larger batches. The results are exhibited in Figure 4.7.

4.3.3.3 Experimentation on the MNIST Dataset: The impact of final BN layer in basic CNNs and FC networks

In order to reproduce equivalent results on a well known benchmark dataset when utilizing different DL architectures, we set up two simple DL architectures: a CNN network with five Conv2D layers and a one-layer (128-node) FC feed forward NN. Then we sampled several pairs of digits (2 vs 8, 3 vs 8, 3 vs 5 and 5 vs 8) from MNIST dataset that are mostly confused in a digit recognition task due to similar patterns in the pixels. During the experiments, the ratio of minority

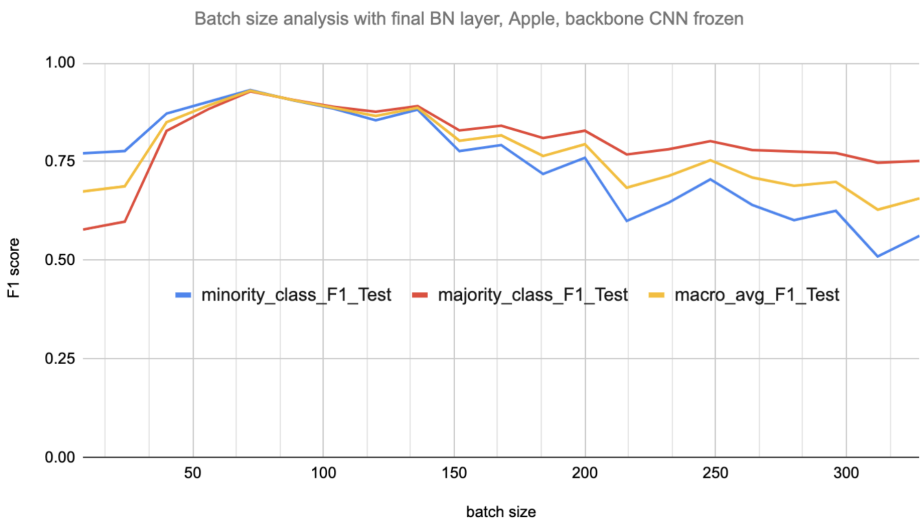


Figure 4.7: The impact of batch size on the F1 test score when the final BN layer added. The highest score is gained when the batch size is 64 and then the accuracy starts declining.

class to majority is kept as 0.1 and experiments with the simple CNN architecture indicates that, after adding the final BN layer, we gain $\sim$ 20% boost in minority class and $\sim$ 10% in majority class. Lower standard deviations across the runs with final BN layer also indicates the regularization effect of using the final BN layer. As a result, we rejected a hypothesis that the performance gain through the final BN layer can be reproduced with another dataset or with much simpler neural architectures. Another important finding is that the final BN layer boosts the minority class' F1 scores only when there is a single majority class. When two majority classes are set, no improvements are evident regardless of the usage of the final BN layer (see 4th and 5th settings in Table 4.7). Therefore, we partially confirm hypothesis (H-5) by showing that the performance gain through the final BN layer can also be achieved in multi-classification settings but the number of majority and minority classes may affect the role of the final BN layer. In the experiments with a one-layer FC network, we enriched the scope and also tested whether using different loss functions and output activation functions would have an impact on model performance using final BN layer with or without another BN

layer after the hidden layer. We observed that adding the final BN layer, softmax output layer and categorical crossentropy (CCE) as a loss function have the highest test F1 scores for both classes. It is also important to note that we observe no improvement even after adding the final BN layer when we use sigmoid activation in the output layer (Table 4.7). Accordingly, we reject hypothesis (H-6).

Table 4.7: Using the same ResNet-34 architecture and skewness (1% vs 99%) and setting up five different configurations across various confusing classes from MNIST dataset, it is clear that adding the final BN layer boosts the minority class F1 scores by 10% to 30% only when there is a single majority class. When we have two majority classes, there is no improvement observed regardless of the final BN layer is used or not. (✓) indicates minority classes.

	Setting-1		Setting-2		Setting-3		Setting-4			Setting-5		
	3 (✓)	8	2 (✓)	8	3 (✓)	5	3 (✓)	5	8	3 (✓)	5 (✓)	8
without final BN	0.19	0.69	0.25	0.70	0.24	0.70	0.06	0.77	0.78	0.28	0.32	0.56
with final BN	0.55	0.76	0.50	0.74	0.54	0.76	0.00	0.77	0.78	0.58	0.59	0.69

Table 4.8: Using one-layer (128 node) NN and the 0.01 skewness ratio, with nine different settings for 3 (minority) and 8 (majority) classes from MNIST dataset (100-epoch). Adding the final BN layer, softmax output layer and CCE as a loss function has the highest test F1 scores for both classes. (CCE - categorical cross entropy, BCE - binary cross entropy, first BN - a BN layer after the hidden layer).

output activation	loss function	first BN layer	final BN layer	class-3 (minority)	class-8 (majority)
sigmoid	BCE			0.17	0.67
softmax	BCE			0.00	0.67
softmax	BCE	✓		0.60	0.78
softmax	CCE			0.67	0.80
sigmoid	BCE		✓	0.05	0.67
softmax	BCE		✓	0.85	0.88
softmax	BCE	✓	✓	0.83	0.87
softmax	CCE		✓	**0.88**	**0.90**
softmax	CCE	✓	✓	0.78	0.85

4.4 Discussion

The metrics at Table 4.2 clearly indicate that after adding the final BN layer, the F1 test score is increased from 0.2942 to 0.9562 for unhealthy Apple, from 0.7237 to 0.9575 for unhealthy Pepper and from 0.5688 to 0.9786 for unhealthy Tomato leaves on average (10 runs). When compared to a network with WL but no final

BN layer, the average improvement was 0.1615, 0.0636 and 0.1115 respectively.

To see what would have happened had we let the models further train, we conducted a 100-epoch training for both models and compared the results. We observed that the test accuracy with the model having a final BN layer quickly rises to 94% within only 6 epochs, but then decreases steadily to 78% within a total of 100 epochs.

By adding a final BN layer just before the output, we normalize the output of the final dense layer, and feed into the softmax layer so that we shrink the gap between class activations (see Figure 4.8). As its name suggests, softmax will always favor the large values when the layer outputs spread over large ranges (exponential normalization). By applying BN to dense layer outputs, the gap between activations is reduced (normalized) and then softmax is applied on normalized outputs, which are centered around the mean. Therefore, we end up with centered probabilities (around 0.5), but favoring the minority class by a small margin.

This shows that DNNs have the tendency of becoming 'over-confident' in their predictions during training, and this can reduce their ability to generalize and thus perform as well on unseen data. Actually, softmax activation leading to an over-confident outputs is already a well-studied phenomenon and observed especially when a neural network is trained with the cross-entropy loss as softmax classifiers tend to output a highly confident prediction [177]. [178] sheds another light on this topic and argues that the deep stack of non-linear layers in between the input and the output unit of a neural network are a way for the model to encode a non-local generalization prior over the input space and it makes output unit to assign nonsignificant probabilities to regions of the input space that contain no training examples in their vicinity.

In addition, large datasets can often comprise incorrectly labeled data, meaning inherently the DNN should be a bit skeptical of the 'correct answer' to avoid being overconfident on bad answers. This was the main motivation of Müller et al. [179] for proposing the *label smoothing*, a loss function modification that has been shown to be effective for training DNNs. Label smoothing encourages the activations of the penultimate layer to be close to the template of the correct class and equally

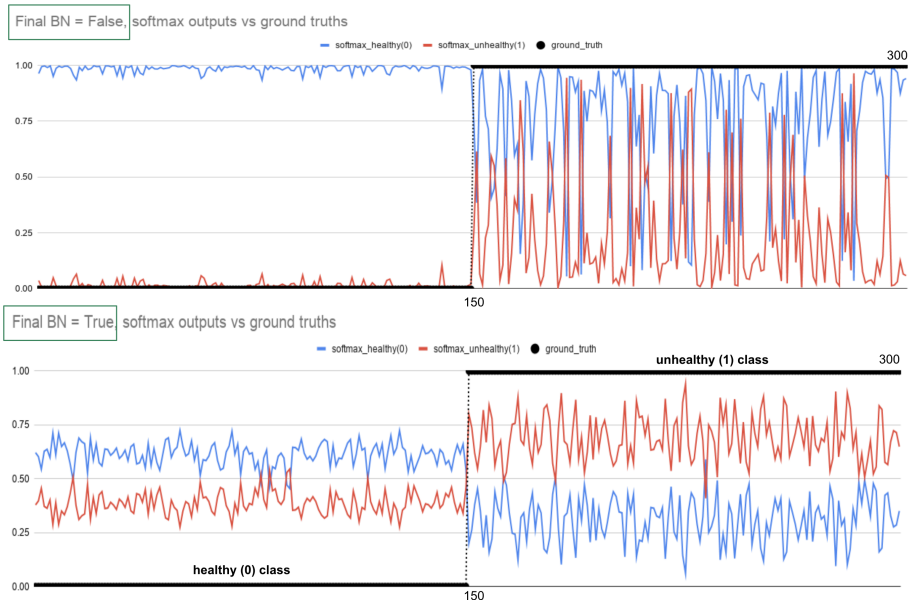


Figure 4.8: The x-axis represents the ground truth for all 150 healthy (0) and 150 unhealthy (1) images in the test set while red and blue lines represent final softmax output values between 0 and 1 for each image. **Top chart (without final BN):** When ground truth (black) is class = 0 (healthy), the softmax output for class = 0 is around 1.0 (blue, predicting correctly). But when ground truth (black) is class = 1 (unhealthy), the softmax output for class = 1 (red points) changes between 0.0 and 1.0 (mostly below 0.5, NOT predicting correctly). **Bottom chart (with final BN):** When ground truth (black) is class = 0 (healthy), the softmax output is between 0.5 and 0.75 (blue, predicting correctly). When ground truth (black) is class = 1 (unhealthy), the softmax output (red points) changes between 0.5 and 1.0 (mostly above 0.5, predicting correctly).

distant to the templates of the incorrect classes [179]. Despite its relevancy to our research, label smoothing did not do well in our study as previously mentioned by Kornblith et al. [180] who demonstrated that label smoothing impairs the accuracy of transfer learning, which similarly depends on the presence of non-class-relevant information in the final layers of the network.

Another observation is that a CNN with final BN is more calibrated when compared to an equivalent network lacking that layer. The calibration principle [181] states that *the probability associated with the predicted class label should reflect its ground truth correctness likelihood* and a model is calibrated if its predicted

outcome probabilities reflect their accuracy. Put it simply, neural networks output "confidence" scores along with predictions in classification. Ideally, these confidence scores should match the true correctness likelihood. For example, if we assign 80% confidence to 100 predictions, then we'd expect that 80% of the predictions are actually correct. If this is the case, we say that the network is calibrated. However, this is not always being observed even in state-of-the-art neural architectures.

We tested the calibration of both networks (with and without final BN) by drawing the reliability diagrams as suggested by Guo et al. [181] and found that when the final BN layer is added, a network has much lower ECE (Expected Calibration Error), i.e., ideal confidence relative to its own accuracy. In effect, a final BN layered network is not 'over-confident' and as a result generalizes and performs better on real world data. This is also confirmed by the fact that the network with a final BN layer has a lower Brier score [182] *(squared error between the predicted probability vector and the one-hot encoded true response)* than without a final BN layer. Our calibration experiments (Figure 4.9.) show that that a network has much lower ECE, i.e. has a more ideal confidence relative to its own accuracy. In effect, a final BN-layered network is not 'over-confident' and as a result generalizes and performs better on unseen datasets.

Figure 4.8 shows that softmax outputs level off around 0.5, as also depicted in Figure 4.3. Since the network loss is optimized by using softmax output value and the ground truth, focusing too much on loss minimization may not be viable when the final BN layer is used in similar settings.

Additionally, we empirically demonstrated that the final BN layer could still be eliminated in inference without compromising the attained performance gain. This finding supports the assertion that adding the BN layer makes the optimization landscape significantly smoother, which in turn renders the gradients' behavior more predictive and stable – as suggested by [158]. We argue that the learned parameters, which were affected by the addition of the final BN layer under imbalanced settings, are likely sufficiently robust to further generalization on the unseen samples, even without normalization prior to the softmax layer.

The observation of locating a sweet spot (10%) for the imbalance ratio, at which

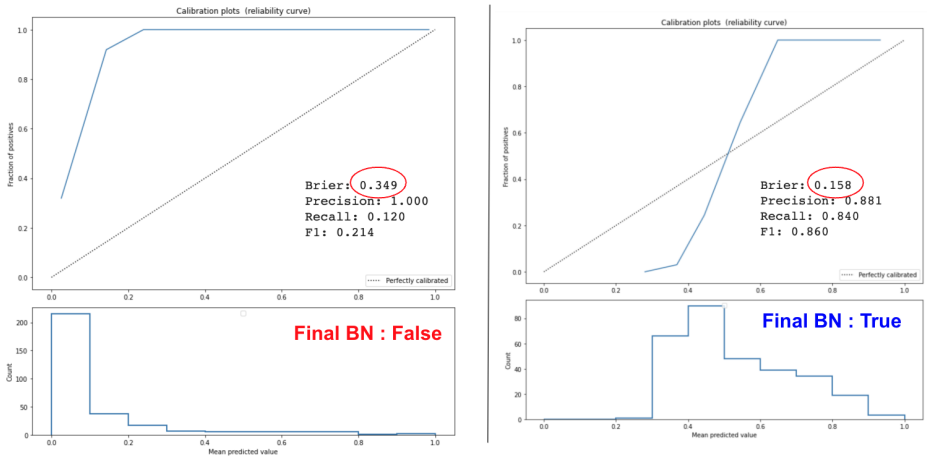


Figure 4.9: The Brier score measures the mean squared difference between the predicted probability assigned to the possible outcomes for item and the actual outcome. Therefore, the lower the Brier score is for a set of predictions, the better the predictions are calibrated. In this chart we see that a network without a final BN layer has a Brier score 0.349 while the one with the final BN has 0.158, indicating a network much better calibrated.

we can utilize the final BN layer, might be explained by the fact that the backbone ResNet architecture is already strong enough to easily generalize on the PV dataset and the model does not need any other regularization once the number of samples from the minority class exceeds a certain threshold. As the threshold found in our experiments is highly related to the DL architecture and the utilized dataset, it is clear that it may not apply to other datasets, but can be found in a similar way.

As discussed before, the BN layer calculates mean and variance to normalize the previous outputs across the batch, whereas the accuracy of this statistical estimation increases as the batch size grows. However, its role seems to change under the imbalanced settings – we found out that a batch size of 64 reaches the highest score, whereas by utilizing larger batches the score consistently drops. As a possible explanation for this observation, we think that the larger the batches, the higher the number of majority samples in a batch and the lesser the chances that the minority samples are fairly represented, resulting in a deteriorated performance.

During our experiments, we expected to see similar behavior with sigmoid activation

replacing softmax in the output layer, but, evidently, the final BN layer works best with softmax activations. Although softmax output may not serve as a good uncertainty measure for DNNs compared to sigmoid layer, it can still do well on detecting the under-represented samples when used with the final BN layer.

Moreover, large datasets can often comprise incorrectly labeled data, meaning inherently the DNN should be a bit skeptical of the 'correct answer' to avoid being overconfident on bad answers. This was the main motivation of Müller et al. [179] for proposing the *label smoothing*, a loss function modification that has been shown to be effective for training DNNs. Label smoothing encourages the activations of the penultimate layer to be close to the template of the correct class and equally distant to the templates of the incorrect classes [179]. Despite its relevancy to our research, label smoothing did not do well in our study as previously mentioned by Kornblith et al. [180] who demonstrated that label smoothing impairs the accuracy of transfer learning, which similarly depends on the presence of non-class-relevant information in the final layers of the network.

4.4.1 Additional Experiments

In addition to the aforementioned experiments, we ran more tests to further support our findings: Turning off the bias at the final dense layer when the final BN layer is added, using SeLU [183] instead of BN in the last layer, freezing and unfreezing the previous BN layers, label smoothing, training a simple CNN from scratch rather than utilizing SOTA architectures.

Also, additional corroboration was achieved over the ISIC Skin Cancer dataset [184] as well as the Wall Crack dataset [185] exhibiting similar patterns of behavior and thus supporting our reported findings. These observations and the code to reproduce the results mentioned in this paper are fully reported in the supplementary material.

4.5 Conclusions

This chapter demonstrated that regardless of freezing or unfreezing the previous layers in the pre-trained modern CNN architectures, putting an additional BN layer just before the softmax output layer has a considerable impact in terms of minimizing the training time and test error for minority classes in a highly

imbalanced dataset (especially in a setting where a high recall is desired for the minority class). Our experiments show that using ResNet34 we can achieve an F1 test score of 0.94 in 10 epochs for the minority class while we could only achieve 0.42 without using the final BN layer. Using VGG19, the same approach increases the F1 test score from 0.25 to 0.96 in 10 epochs. Using VGG19, the same approach increases the F1 test score from 0.25 to 0.96 in 10 epochs. As evident from Table 4.2, upon adding the final BN layer the F1 test score is increased from 0.2942 to 0.9562 for the unhealthy Apple minority class, from 0.7237 to 0.9575 for the unhealthy Pepper and from 0.5688 to 0.9786 for the unhealthy Tomato when WL is not used (all are averaged values over 10 runs). When compared to a network with WL but without a final BN layer, the averaged improvements were 0.1615, 0.0636 and 0.1115 respectively. We showed that the highest gain in test F1 score for both classes (majority vs. minority) is achieved just by adding a final BN layer, resulting in a more than three-fold performance boost on some configurations.

We also argueed that trying to minimize validation and train losses may not be an optimal way of getting a high F1 test score for minority classes. We presented that having a higher train and validation loss but high validation accuracy would lead to higher F1 test scores for minority classes in less time. That is, the model might perform better even if it is not confident enough while making a prediction. So, having a confident model (lower loss, higher accuracy) may favor the majority class in the short run and ends up with lower F1 test scores for minority class due to the conjecture that it learns to minimize error in majority class regardless of class balance. Since BN (like dropout) adds stochasticity to the network, and the network learns to be robust to this stochasticity during the entire training, it was expected that the final BN layer would be detrimental for the network to cope with the stochasticity right before the output [167]. However, we observed the contrary in our experiments.

This chapter also showed that lower values in the softmax output may not necessarily indicate 'lower confidence level', leading to another discussion why softmax output may not serve as a good uncertainty measure for DNNs. In classification, predictive probabilities obtained at the end of the pipeline (the softmax output) are often wrongly interpreted as model confidence. As we have clearly shown, a model can be uncertain in its predictions even when having a high softmax output [186].

In the experiments mentioned in this chapter, we noticed that the performance gain after adding the final BN layer in highly imbalanced settings could still be achieved after removing this additional BN layer during inference; in turn enabling us to get a performance boost with no additional cost in production. Then we explored the dynamics of using the final BN layer as a function of the imbalance ratio within the training set, and found out that the impact of final BN on highly imbalanced settings is the most apparent when the ratio of minority class to the majority is less than 10%; there is hardly any impact above that threshold.

We also ran similar experiments with simpler architectures, namely a basic CNN and a single-layered FC network, when applied to the MNIST dataset under various imbalance settings. The simple CNN experiments exhibited a gain of $\sim 20\%$ boost per the minority class and $\sim 10\%$ per the majority class after adding the final BN layer. In the FC network experiments, we observed improvements by 10% to 30% only when a single majority class was defined; no improvements were evident for two majority classes, regardless of the usage of the final BN layer. While experimenting with different activation and cost functions, we found out that using the final BN layer with sigmoid activation had almost no impact on the task at hand.

Overall, our study has shed light on the unexpected impact of the final BN layer in simpler neural networks while dealing with imbalanced datasets. This discovery presents exciting opportunities for further investigation in the future, as it requires a more comprehensive analysis to fully understand its implications. Moving forward, we suggest that future researchers explore the generalization of our findings to a wide range of neural models, utilizing a combination of softmax activation or an appropriate loss function for use in imbalanced image classification problems.

Additionally, we recommend examining the potential implications of our findings in real-world scenarios, such as improving the accuracy of image classification in medical diagnosis or object recognition in autonomous vehicles. We believe that exploring these avenues will not only advance our understanding of neural networks, but also have practical implications for a variety of fields.

To sum, the overall contributions of this chapter have been the following compo-

nents:

- Putting an additional BN layer just before the output layer has a considerable impact in terms of minimizing the training time and test error for minority classes in highly imbalanced image classification tasks. Our experiments show that the initial F1 test score increases from the 0.29-0.56 range to the 0.95-0.98 range for the minority class when we added a final BN layer just before the softmax output layer.

- Trying to minimize validation and train losses may not be an optimal way for getting a high F1 test score for minority classes in binary anomaly detection problems. We illustrate that having a higher training and validation loss but high validation accuracy leads to higher F1 test scores for minority classes in less time and using the final BN layer has a calibration effect. That is, the network might perform better even if it is not 'confident' enough while making a prediction. We will also argue that lower values in softmax output may not necessarily indicate 'lower confidence level', leading to another discussion about why softmax output is not a good uncertainty measure for deep learning (DL) models.

- The performance gain after adding the final BN layer in highly imbalanced settings could still be achieved after removing this additional BN layer during inference; in turn enabling us to get a performance boost with no additional cost in production.

- There is a certain threshold for the ratio of the imbalance for this specific PV dataset, upon which the progress is the most obvious after adding the final BN layer.

- The batch size also plays a role and significantly affects the outcome.

- We replicated the similar imbalanced scenarios in MNIST dataset, reproduced the same BN impact, and furthermore demonstrated that the final BN layer has a considerable impact not just in modern CNN architectures but also in simple CNNs and even in one-layered feed-forward fully connected (FC) networks.

- We illustrate that the performance gain occurs only when there is a single majority class and multiple minority classes; and no improvement observed regardless of the final BN layer when there are two majority classes.
- We argue that using the final BN layer with sigmoid activation has almost no impact when dealing with a strongly imbalanced image classification tasks.

Chapter 5

The Role of Self-Supervised Learning

5.1 Self-Supervised Learning

5.1.1 Motivation

CV models have typically been trained using supervised learning, a process in which humans provide labels for images in order to help the model recognize patterns. This can be time-consuming, as it requires human annotators to assign class labels or draw bounding boxes around objects in the images. An alternative approach is self-supervised learning (SSL), in which the model is able to learn from the data itself, without the need for human-provided labels.

This is often achieved by applying different image transformations or crops to the same image, and then using the model to learn that these modified images still contain the same visual information. During the SSL training process (Figure 5.1), the augmented version of the original sample is considered as a positive sample, and the rest of the samples in the batch/dataset (depends on the method being used) are considered negative samples. Next, the model is trained in a way that it

learns to differentiate positive samples from the negative ones [187]. While doing so, the model learns quality representations of the samples and is used later for transferring knowledge to downstream tasks.

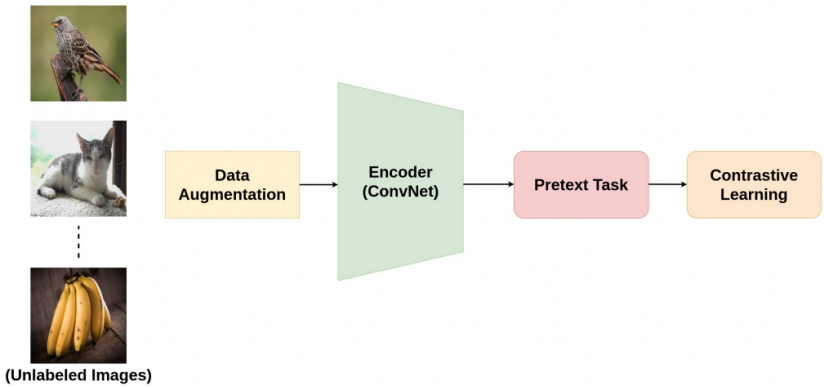


Figure 5.1: Contrastive learning pipeline for self-supervised training (source: [187])

This idea is advocated by an interesting experiment conducted by Epstein in 2016 [188], where he asked his students to draw a dollar bill with and without looking at the bill (from memory) (Figure 5.2). Evidently, the drawing made in the absence of the dollar bill is quite primitive compared with the drawing made from an exemplar, even though the students have seen a dollar bill thousands of times. The results from the experiment show that the brain does not require complete information of a visual piece to differentiate one object from the other. Instead, only a rough representation of an image is enough to do so [187].

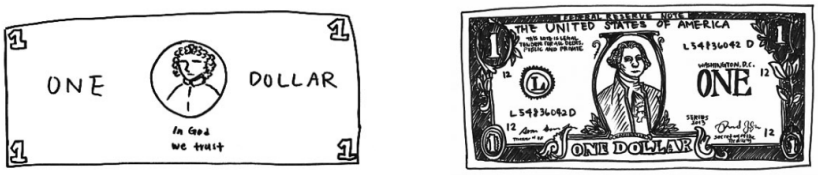


Figure 5.2: Epstein dollar bill experiment: Drawing a dollar bill with and without looking at the bill (source: [188])

This training process allows the model to learn a latent representation (a vector output) for the same objects, which can be useful for downstream tasks such as object detection and semantic segmentation. Transfer learning can then be applied to this pre-trained model, allowing it to be fine-tuned on a smaller labeled dataset to perform specific tasks. As we'll be illustrating clearly in the following sections, models trained with self-supervised learning (SSL) tend to generalize better than their supervised counterparts for transfer learning (see Figure 5.3)

5.1.2 Background

The capacity to discover structure in the world without someone explicitly providing supervision/teaching is considered a human-level capability. Likewise, learning about the structure of data leads to useful representations about the world. Ideally, we'd want neural networks to learn initial structures which have wide applicability and to do so on their own without human annotated labels. On the other hand, supervised learning requires vast amounts of carefully annotated and curated data that actually increases the amount of tedious and laborious jobs in many ways. It's almost impossible to label everything in the world with every possible useful label if we want to achieve a decent generalization power that would lead to zero-shot learning capabilities. We need orders of magnitude more data, to which labeling is going to be very expensive.

The proposed solution to this issue is usually applying a transfer learning, a well-known technique in CV that involves using pre-trained deep convolutional neural networks (DCNNs) as a starting point to learn a new task. These large models are trained on massive supervised datasets, such as ImageNet, and their features have been found to adapt well to new problems. Transfer learning is often used when annotated training data is scarce and involves taking the pre-trained weights of a model, adding a new classifier layer on top, and retraining the network. It has been shown that using pre-trained weights to initialize a model for a new task tends to result in faster learning and higher accuracy compared to training from scratch with random initialization.

However, transfer learning relies on models trained on supervised datasets, which can be costly to create due to the need for annotation. In contrast, unsupervised

data is abundant and can be used to learn representations that can be used to improve supervised models. This makes it a potentially valuable approach for improving the efficiency and effectiveness of transfer learning. Also known as unsupervised learning, unsupervised representation learning is concerned with addressing the following issue: learning good representations from unlabeled data.

Unsupervised representation learning, which involves training a model to learn useful representations of data without the need for labeled examples, has the potential to unlock a number of applications that current transfer learning has not been able to address. However, it has historically been a more difficult problem than supervised representation learning. One example of this is in the task of breast cancer detection, where the best solutions currently rely on ImageNet pre-trained models as a starting point. Despite the significant differences between breast cancer images and regular ImageNet samples, transfer learning still proves effective to some extent in this context. This is because most supervised datasets for breast cancer detection are smaller and have less variability than common CV supervised datasets. However, there is a large number of non-annotated slide images of breast cancer available, and if good representations could be learned from this unsupervised data, it could help to improve specific downstream tasks that have limited annotated data. Fortunately, visual unsupervised representation learning has shown promise, particularly with the use of contrastive-based techniques. These techniques are now able to learn visual representations that are similar to those learned using supervised methods in some self-supervised benchmarks.

There is often confusion surrounding the concepts of unsupervised learning and self-supervised learning. While self-supervised learning can be considered a subcategory of unsupervised learning because it does not involve manually-provided labels, unsupervised learning is more specifically focused on identifying specific patterns in data, such as clustering, community discovery, or anomaly detection. On the other hand, self-supervised learning aims to reconstruct or predict missing information, which is more similar to the goals of supervised learning. In this sense, self-supervised learning can be seen as a hybrid approach that combines elements of both supervised and unsupervised learning [31].

The self-supervised training strategy in CV has so far been inspired by the approach

in natural language processing (NLP) in a sense of pre-training on a vast amount of training data and then fine-tuning on a target data set [189]. Owing to the latest breakthrough in 'attention' based neural network architectures [190], namely 'transformers', no one would argue that the key ingredient of success in the NLP world recently has been the self-supervised pre-training. The pre-training tasks use vast amounts of data available online to create a learning signal and this scale is much better than supervised learning and it doesn't deflate away any of the rich semantic information in the images by projecting onto a single label selected from only a handful of possible categories. Given different representations of the same image we want to learn a representation space where representations from the same image are close together and at the same time far apart from representations of other images. This simple idea is remarkably based on nothing other than similarity, which eventually leads to powerful representations.

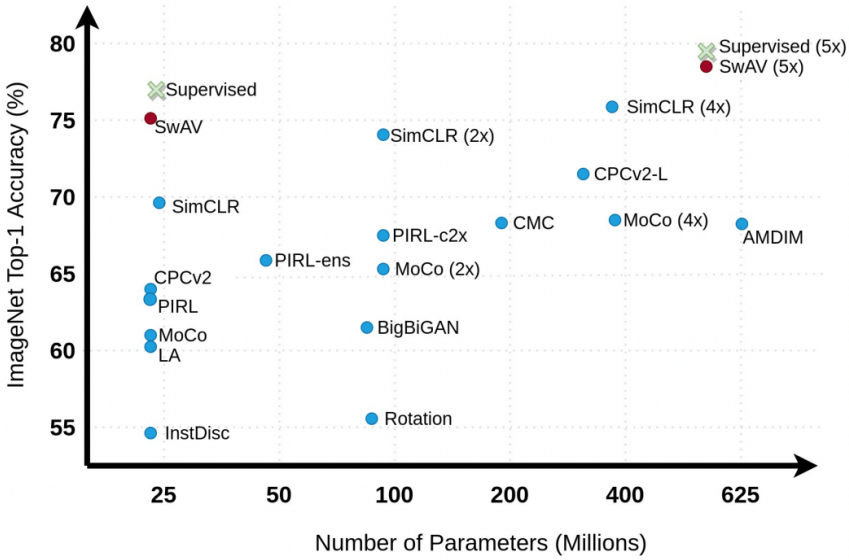


Figure 5.3: Self-supervised representation learning performance on ImageNet top-1 accuracy under linear classification protocol. The self-supervised learning's ability on feature extraction is rapidly approaching the supervised method. (source: [187])

SSL models can be grouped under four categories:

- Generative (autoregressive models (predicting the next token/pixel), non-autoregressive models (masking a token/pixel and predicting the masked token/pixel) flow-based models, auto-encoding (AE) models, and hybrid generative models)
- Predictive tasks (predicting the relative location of the image patches, predicting whether the next fragment is the next sentence, solving the puzzles created from shuffled patches of an image, predicting the cluster id of each sample, predicting the image rotation angle, etc.)
- Contrastive (MoCo, SimCLR, SimSiam etc.)
- Generative-Contrastive (adversarial) (GAN, VQ-VAE2, BiGAN etc.)

5.1.3 Contrastive Self-Supervised Learning

In this dissertation, we will focus more on contrastive SSL approaches. Contrastive learning involves the creation of positive and negative training sample pairs based on the characteristics of the data. The goal is for the model to learn a function that assigns high similarity scores to positive samples and low similarity scores to negative samples. Therefore, it is crucial to properly generate the samples in order to ensure that the model can effectively learn the underlying features and structures of the data.

Forcing the model to generate the supervisory signals out of the data itself, the machine supervision is the process of creating a label but the human supervision is reduced to the process of selecting suitable data sets and data augmentations. We're still adding prototypical human knowledge into the model via the back door but with significantly less onus on the humans. In other words, we're effectively letting the machine itself discover structures and categories by predicting unobserved or hidden relationships either in time or in space. We see similar examples around the same idea in the literature. Recent advances in large language models such as ChatGPT and GPT-4 have demonstrated the potential of machines to discover patterns and relationships in data on their own, paving the way for new approaches to machine learning and artificial intelligence that rely less on human supervision. As we continue to explore these exciting new possibilities, it's clear that the role of

humans in machine learning will continue to evolve and transform in the years to come.

Self-supervised learning in NLP, particularly with BERT [191] and GPT [192], has demonstrated significant success, prompting researchers to explore the application of similar generative approaches in other domains such as image and speech recognition. However, generating masked inputs for image data presents more challenges compared to text data, as it is more difficult to selectively mask a large number of image pixels (it is easier to choose a limited amount of text tokens).

With the help of SSL, in CV we're trying to learn the fundamental property of an image that you should be able to recognize irrespective of many common distortions of images. The idea of using data augmentations to vary individual data points and then differentiating all of those variations from other data points is not new by any means. Exemplar networks for example cast the representation learning problem as a classification problem with every image in the data set representing its own class. Later the idea is refined and the problem is converted from a parametric to a non-parametric classification by using the data points themselves as classification weights.

The idea of Noise Contrastive Estimation (NCE) from the discipline of learning word vectors and NLP is highly used in SSL based models. NCE [193] is a way of learning a data distribution by comparing it against a noise distribution. This allows us to cast an unsupervised problem as a supervised problem. It is similar to Negative Sampling, which is an approximation mechanism that was invented to reduce the computational cost of normalizing network outputs by summing over the entire vocabulary. Negative Sampling [194] is a backbone of word embeddings concept in NLP (learning vector embeddings of words to help with Natural Language tasks) and it aims at maximizing the similarity of the words in the same context and minimizing it when they occur in different contexts. However, instead of doing the minimization for all the words in the dictionary except for the context words, it randomly selects a handful of words depending on the training size and uses them to optimize the objective.

One of the most popular self-supervised frameworks with this simple concept

of NCE, SimCLR [195] delivered another push in performance by dramatically simplifying the architecture and the data augmentations used. Now in the early days of contrastive SSL such as FaceNet and SimCLR, we had explicit positives and negatives. The models would try to distinguish different augmented images by classifying which image it was.

Contrastive learning in the field of CV involves using data augmentation to generate positive sample pairs from a single original image, and using two distinct images as negative sample pairs. Two key factors that can affect the effectiveness of this approach are the strength of the data augmentation and the selection of negative sample pairs. If the data augmentation is too strong, such that the two augmented samples bear no relationship to one another, the model will not be able to learn useful information. Similarly, if the data augmentation is too weak, the model will also not be able to learn effectively. In terms of selecting negative sample pairs, it is important to avoid randomly assigning two images as negative pairs if they are likely to belong to the same class, as this can introduce conflicting noise to the model. At the same time, the negative pairs should not be too easy to distinguish, as this can prevent the model from learning the underlying features and structures of the data.

This approach of contrastive learning, i.e., learning representations by enforcing similar elements to be equal and dissimilar elements to be different, made the strong assumption that everything else that isn't within this particular class is dissimilar. At the same time, this approach was also quite flawed because training a classifier to distinguish between all the possible pairs of images didn't scale very well with the number of images. SimCLR also produced this really clever idea of a projection head where the representations we compare in the objective are not the ones we actually use downstream. This means that we can effectively learn a broad similarity but maintain fine grained representations. The projectors may have as many parameters as the backbones themselves.

It's a really interesting question if we just want to turn an image into a vector that we could then train a linear classifier on top of what is the best way of doing that. The "bootstrap your own latent", or BYOL method [196], moved away from comparing different images and instead focused on metric learning.

Features are trained by comparing them to a momentum encoder and using a much smaller batch size. It turned out that the momentum encoder wasn't even that useful. What's clear from looking at the self-attention masks is that the self-supervised representations are far richer than the supervised ones even though the supervised ones typically have slightly better accuracy. Supervised classification is like cheating on an exam by knowing what the questions will be in advance and all the model needs to know is how to pass that particular exam. This is the so-called shortcut rule where we optimize on a particular metric at the cost of everything else. We already know from the No Free Lunch theorem [197] that such highly specific accuracy comes at the cost of generalization. Self-supervised learning can bypass such specific and narrow objectives that perform well at specific downstream tasks and instead build much more broadly applicable and general purpose representations.

Data augmentation is the secret and the most important ingredient in making self-supervised learning work so well. The data augmentation that we need for self-supervised learning is different from the data augmentation that we need for supervised learning. When we have contrastive learning, we have this problem which is that if there's only one dominant feature in your data that can be used to perform the contrastive task. That would be the only feature that the network would ever learn. So with augmentation we're somewhat making the task harder so that the network actually has to learn many different kinds of features.

In a nutshell, the data augmentation techniques that we do are different for SSL compared to supervised learning. For instance, aggressive color distortion is quite important in self-supervised learning. This is because in the self-supervised setting where we're comparing different augmented representations from the same image it would be easy for the model to overfit on the color histogram instead of learning the richer information. If we think about it, not all trees in nature have the same color histogram but in SSL all of the individual images will have the same color histogram. Data augmentation also serves as a proxy for what might happen. In real life we're an agent interacting with the environment going around all of these different places and we see objects in a variety of different circumstances and we know different transformations being applied to our viewpoint.

In this section, we will briefly explore the popular SSL algorithms to give a rough idea about how each one of them tackles the problem of extracting the signals from unlabelled data. You can find a detailed survey of SSL methods in [187].

- SimCLR [195]
- MoCo [198]
- SimSiam [199]
- BYOL [196]
- SwAV [200]
- Barlow Twins [201]
- DINO: [202]

BYOL and Barlow Twins in this list are considered as non-contrastive SSL methods.

5.1.3.1 SimCLR: A Simple Framework for Contrastive Learning of Visual Representations

There has been a proliferation of SSL approaches for learning image representations in recent years, with each method showing incremental improvements over previous ones. However, the performance of these methods has generally remained inferior to that of supervised learning methods. This changed with the publication of the SimCLR paper [195], which demonstrated not only superior performance over previous state-of-the-art self-supervised learning methods, but also outperformed supervised learning methods on ImageNet classification when using a larger model architecture.

The SimCLR framework is based on a straightforward concept: a single image is transformed through various random modifications to generate a pair of augmented images. These images are then processed through an encoder, resulting in representations that are further transformed by a non-linear fully connected layer. The objective of the framework is to maximize the similarity between the two resulting representations for a given image (see Figure 5.4).

The training flow can be described as follows:

1. Obtain an input image
2. Apply two random augmentations to the image, including transformations such as rotations, changes in hue/saturation/brightness, zooming, and cropping, etc. (see [195] for the entire list and ranges).
3. Use a deep neural network (preferably a convolutional one, such as ResNet50) to generate image representations (embeddings) of the augmented images.
4. Utilize a small, fully connected linear neural network to project the embeddings into another vector space.
5. Compute the contrastive loss and apply backpropagation through both networks. The contrastive loss decreases when the projections derived from the same image are similar (e.g. have a high cosine similarity).

The contrastive loss decreases when the projections of augmented images that were derived from the same input image are similar. For two augmented images (i) and (j) that were generated from the same input image, the contrastive loss for (i) tries to identify (j) among the other images in the same batch.

5.1.3.2 MoCo: Momentum Contrast for Unsupervised Visual Representation

MoCo was first presented in [198] in 2020 and further improved to MoCo v2 in [203]. There is another iteration called MoCo v3 in [204] which introduces a fundamental adaptation of the training process and the model architecture. In the following paragraphs, We will first discuss the original ideas presented in the original implementation [198], followed by a review of the improvements made in subsequent versions as the method has been continually refined.

In the MoCo paper, the unsupervised learning process is framed as a dictionary look-up: Each view or image is assigned a key, just like in a dictionary. This key is generated by encoding each image using a convolutional neural network, with the output being a vector representation of the image. Now, if a query is presented to

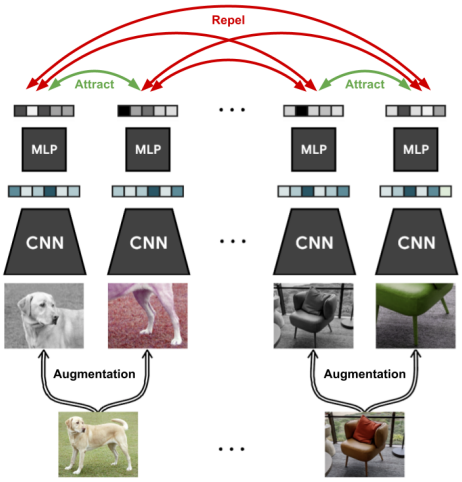


Figure 5.4: An illustration of the SimCLR training procedure. SimCLR employs contrastive learning to optimize the consistency between two augmented versions of the same image. The aim is to increase the similarity between these two augmented representations of the same image. (source: https://simclr.github.io/)

this dictionary in the form of another image, this query image is also encoded into a vector representation and will belong to one of the keys in the dictionary, the one with the lowest distance.

In the original study, the unsupervised learning process is described as a dictionary look-up: Each view or image is assigned a key, similarly to the way words are assigned definitions in a dictionary. The key is generated by encoding each image using a convolutional neural network, which produces a vector representation of the image. If a query, in the form of another image, is presented to this dictionary, it is also encoded into a vector representation and will be associated with the key in the dictionary that has the lowest distance from it.

The training process works as follows (see Figure 5.5): A query image is selected and processed by the encoder network to compute the encoded query image (q). The goal of the model is to learn to differentiate between a large number of different images, so the encoding of the query image is compared to not just one mini-batch of encoded key images, but to multiple mini-batches. To do this, the MoCo algorithm maintains a queue of mini-batches that are encoded by the momentum encoder

network. When a new mini-batch is selected, its encodings are added to the queue and the oldest encodings are removed. This allows the model to query from a much larger dictionary, decoupling the dictionary size (represented by the queue) from the batch size. If the encoding of the query image matches a key in the dictionary, it is determined that the two views are from the same image (e.g. different crops of the same image). If the encoding of the query image is found to be similar to a key in the dictionary, it can be inferred that the two views are from the same image (e.g. different crops of the same image).

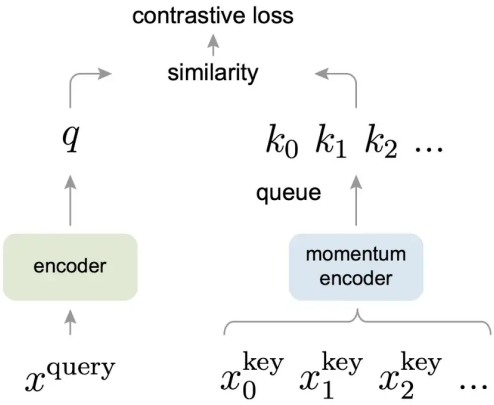


Figure 5.5: The encoder to compute the encoded query image is depicted on the left, and the queue of mini-batches of keys and the momentum encoder is on the right [198].

This loss function enables the model to learn to assign smaller distances to views from the same image and larger distances to views from different images. The model is trained using backpropagation through the encoder network. The authors aim to obtain a consistent encoding for each image using the momentum encoder. To prevent the weights from becoming frozen and not reflecting the progress of the model's learning, they propose using a momentum update instead of copying the encoder's weights. This is achieved by setting the momentum encoder network to be updated using a momentum-based moving average of the query encoder for each training iteration.

In MoCo v2 [203], the authors incorporate some of the ideas from the SimCLR paper into their model. Previously, for training a linear classifier after pre-training

MoCo, they had added a single linear layer to the model. In MoCo v2, this layer has been replaced with an MLP projection head, resulting in improved classification performance. Additionally, stronger data augmentations similar to those used in SimCLR have been introduced, including the addition of blur and stronger color distortion to the v1 augmentations. In the latest version, MoCo v3, a different training process is adopted to improve the architecture.

5.1.3.3 SimSiam: Simple Siamese Representation Learning

In previous approaches, the use of large numbers of negative sample pairs, which required large batch sizes, resulted in high computational and memory demands. Some prior approaches addressed this issue by employing momentum encoders as a workaround to avoid the need for large batch sizes. SimSiam aims to eliminate these requirements and create a simple SSL framework.

SimSiam is a proposed simple Siamese network that is able to learn meaningful representations without using negative sample pairs, large batches, or momentum encoders. The Siamese structure is a key factor in the overall success of the considered baselines and a stop-gradient operation plays a crucial role in preventing collapse. It does not require a large batch size in order to operate, unlike SimCLR, and performs better than SimCLR in most of the cases.

[205] says that the performance actually degrades as the number of negatives is increased for a fixed batch size and using fewer negatives can lead to a better signal-to-noise ratio for the model gradients. This could explain the SimSiam's improved performance with respect to SimCLR since large negative samples or large batch sizes to ensure large negative samples are not needed in SimSiam.

SimSiam follows a standard contrastive learning setup in which two networks with shared weights are presented with two different augmented views of the same image, and an MLP projection head transforms one view to match the other. Notably, the contrastive loss is symmetrical: the output of one of the networks is treated as constant (using stop-gradient), and the output of the second network is projected to match it. The roles of the two networks are then reversed, with the output of the first network projected and matched to the "constant" output of the second network. The two losses are averaged, and each of the networks receives gradient

for the loss term, with stop-gradient not applied to its output.

In a nutshell, given an image, we create two augmented views and process these two versions with the same encoder network (a backbone plus a projection MLP). Then, we maximize the similarity at the end by minimizing the negative cosine similarity to optimize the parameters.

5.1.3.4 BYOL: Bootstrap Your Own Latent

Contrastive methods are less expensive as described by the authors of BYOL [196]: "Contrastive approaches avoid a costly generation step in pixel space by bringing representation of different views of the same image closer ('positive pairs'), and spreading representations of views from different images ('negative pairs') apart."

Having to compare each sample to many other negative samples is problematic, because it introduces instabilities into the training, and reinforces systematic biases from the dataset. Here is how this phenomenon es clearly explained in [196]: "Contrastive methods are sensitive to the choice of image augmentations. For instance, SimCLR does not work well when removing color distortion from its image augmentations. As an explanation, SimCLR shows that crops of the same image mostly share their color histograms. At the same time, color histograms vary across images. Therefore, when a contrastive task only relies on random crops as image augmentations, it can be mostly solved by focusing on color histograms alone. As a result the representation is not incentivized to retain information beyond color histograms."

Unlike other contrastive learning methods, BYOL achieves state-of-the-art performance without requiring negative samples. Like a siamese network, BYOL uses two identical encoder networks, referred to as the online and target networks, to obtain representations and minimizes the contrastive loss between the two representations.

The goal of BYOL is to ensure that similar samples have similar representations, while contrastive learning aims to ensure that dissimilar samples have dissimilar representations. Training with BYOL is more efficient, as it does not require negative sampling and only samples each training example once per epoch. This can lead to better generalization of the model, as it is less sensitive to systematic biases

in the training dataset. BYOL minimizes the distance between the representation of each sample and a transformed version of that sample, such as through translation, rotation, or blurring.

Since BYOL does not require negative sampling, the training process is more efficient as we only sample each training example once per epoch. The negative counterparts can be ignored altogether. In return, the model is less sensitive to systematic biases in the training dataset, leading to better generalizations on unseen examples.

5.1.3.5 SwAV: Swapping Assignments between multiple Views of the same image

As explained before, contrastive learning methods involve the direct comparison of features from different transformed versions of the same images. In contrast, SwAV does not directly compare image features. Instead, it involves the creation of intermediate targets by assigning image features to prototype vectors and solving a "swapped" prediction problem, in which the targets are altered for two different views of the same image. Figure 5.6 illustrates the difference between contrastive instance learning and swapping assignments between the views.

SwAV is more efficient because it does not require a large memory bank or an auxiliary momentum network. Instead of directly comparing features, it clusters the data and enforces consistency between cluster assignments for different augmentations of the same image simultaneously. In other words, it uses a "swapped" prediction mechanism where the cluster assignment of a view is predicted from the representation of another view. This method can be trained with large and small batches and can handle an unlimited amount of data.

One of the important contributions of the original SwAV paper is a new data augmentation strategy called 'multi-crop', that uses a mix of views with different resolutions in place of two full-resolution views, without increasing the memory or compute requirements. Hence, not only two larger crops are created, but also up to 4 smaller crops are taken. This is one of the keys for why SwAV can boost its performance in comparison to other approaches [200].

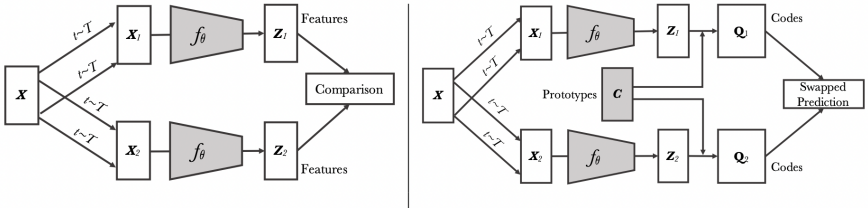


Figure 5.6: In contrastive learning methods applied to instance classification, the features from different transformations of the same images are compared directly to each other. In SwAV, the codes obtained from one data augmented view are predicted using the other view. Thus, SwAV does not directly compare image features [200].

5.1.3.6 Barlow Twins: Self-Supervised Learning via Redundancy Reduction

Recent research in SSL aims to learn embeddings that are invariant to distortions of the input sample. This is typically accomplished by applying augmentations to an input sample and attempting to make their representations as similar as possible. However, a common issue is that there are often trivial constant solutions that the network can rely on, resulting in no meaningful learning.

Similarly to other SSL methods, Barlow twins, aims at learning representations (embeddings) that are invariant to input distortions. An important detail is that the encoder is actually made up of an encoder network followed by a "projector" network. The encoder executes feature extraction while the projector designs the embedding's high dimensional space. Representations learned are used to train classifiers on downstream tasks, while embeddings are fed into the loss function to train the model.

The objective function in the Barlow Twins method is designed to prevent collapse by measuring the cross-correlation matrix between the outputs of two identical networks fed with distorted versions of a sample. This causes the embedding vectors of distorted versions of a sample to be similar while minimizing redundancy between the components of these vectors. This method does not require large batches or asymmetry between the network twins, such as a predictor network, gradient stopping, or a moving average on weight updates. Interestingly, it benefits

from very high-dimensional output vectors. It outperforms previous methods on ImageNet for semi-supervised classification in the low-data regime and performs comparably to the current state-of-the-art for ImageNet classification with a linear classifier head and for transfer tasks of classification and object detection. The Barlow twins method either outperforms or performs comparably to most other self-supervised learning architectures on a variety of computer vision downstream tasks [201].

5.1.3.7 DINO: Emerging Properties in Self-Supervised Vision Transformers

DINO works by interpreting self-supervision as a special case of self-distillation, where no labels are used at all. The idea is to train a student network by simply matching the output of a teacher network over different views of the same image. The authors of DINO [202] identified two components from previous self-supervised approaches that are particularly important for strong performance on the Vision Transformer (ViT) – the momentum teacher (MoCo) and multi-crop training (SwAV) – and integrated them into DINO framework. The resulting model achieves state-of-the-art performance above all previously proposed self-supervised systems, clearly depicting the potential of ViTs for self-supervised learning.

In DINO, there is a teacher and student network, both having the same architecture, a Vision Transformer (ViT). The teacher is a momentum teacher, which means that its weights are an exponentially weighted average of the student's. During training, different crops of one image are taken. Small crops are called Local views and large crops are called Global views. All crops are passed through the student, while only the global views are passed through the teacher. Additionally, random augmentations are applied on the views to make the network more robust. The Cross entropy loss is applied to make the student's distribution match the teacher's, allowing the student network to have the same proportions of features as the teacher. The teacher, having a larger context, predicts more high-level features, which the student must also match.

To ensure the teacher's knowledge is not lost, the stop-gradient operator is applied on the teacher to propagate gradients only through the student. The teacher param-

eters are updated with an exponential moving average of the student parameters. This process is aimed at making the network address a classification problem such that it can learn meaningful global representations from local views. To prevent mode collapse two techniques are used: Centering and Sharpening. In centering, the teacher's raw activations have their exponentially moving average subtracted from them. In sharpening, temperature is applied to the softmax to artificially make the distribution more peaked. DINO learns a significant amount about the visual world by identifying object parts and shared characteristics across images, resulting in a feature space with a structured and interpretable organization of similar categories. This allows for efficient k-NN classification without the need for fine tuning or learning classifiers. When compared to state-of-the-art SSL techniques on the task of ImageNet classification, DINO consistently demonstrates superior performance across different throughput regimes. The original DINO paper [202] also shares some interesting findings and observations regarding the zero shot object segmentation capacity of DINO. DINO has shown to have understood object semantics so well that its attention maps look like segmentation masks. Figure 5.7 shows such an example in which a ViT network trained with DINO with no supervision automatically learns class-specific features and able to segment the objects better than a network that is trained with full supervision.

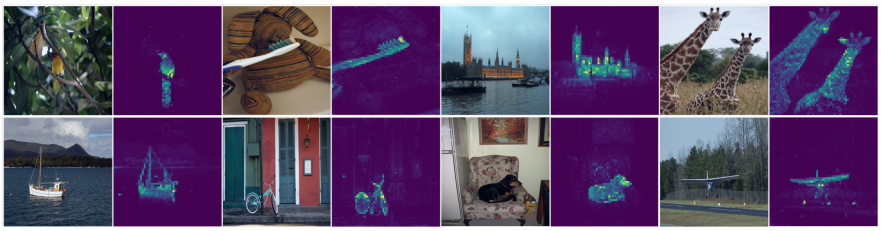


Figure 5.7: Self-attention from a ViT with 8×8 patches trained with no supervision automatically learns class-specific features leading to unsupervised object segmentations [202].

Other than the ones covered above, there are many other SSL algorithms proposed so far. Even though the fundamental idea behind each of these methods looks similar, there might be some nuances in their implementation due to the tasks they're trying to solve. Figure 5.8 shows the fundamental differences between siamese based SSL algorithms in one visual.

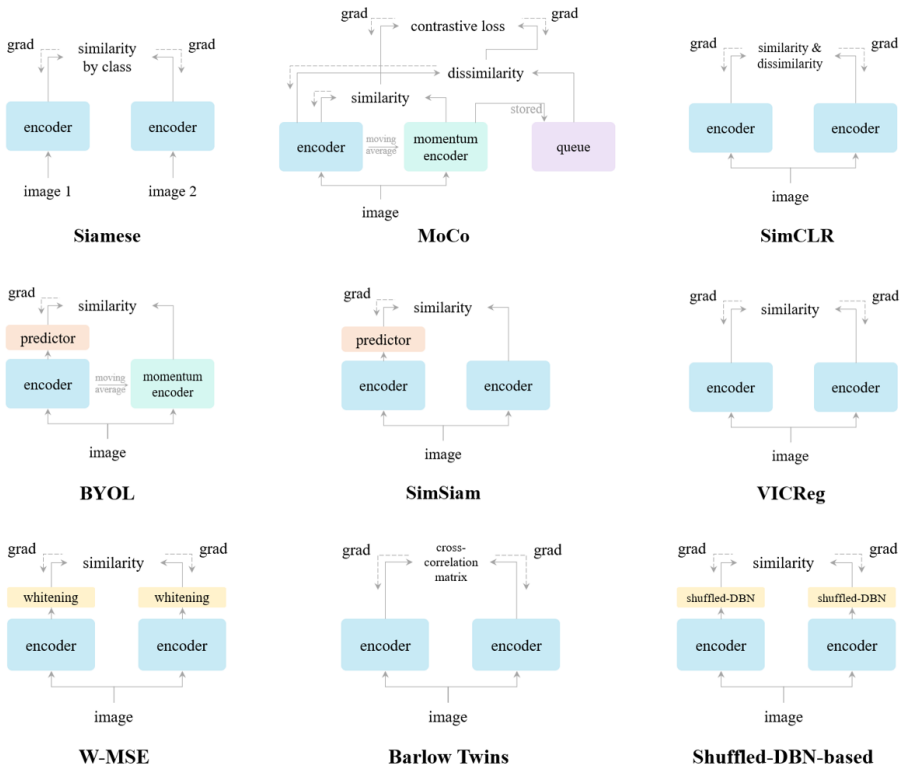


Figure 5.8: Comparison on Siamese architecture-based SSL frameworks. The encoder includes all layers that can be shared between both branches. The dash lines indicate the gradient propagation flow. Therefore, the lack of a dash line denotes stop-gradient [206].

5.2 Preliminary: SSL applied to small subsets of Plant Village data

In order to assess the effectiveness SSL on small data, we run some experiments with SimCLR on Plant Village (PV) dataset that we explored in detail in chapter 4. Here we applied the following steps:

1. Train a supervised image classifier with ResNet34 on the entire PV dataset **using the labels**, drop the fully connected (FC) classifier (head) layers, use the backbone network as a feature extractor, collect feature embeddings of a small sample of PV dataset (Apple class, 10 healthy, 10 unhealthy), train a

simple classifier model (backbone network plus a couple of FC dense layers as a head classifier) on this small dataset to classify healthy and unhealthy classes, and then test on a comparatively larger dataset (Apple class, 150 healthy, 150 unhealthy).

2. Train a self-supervised model with SimCLR (backbone ResNet34) model on the entire PV dataset **without using the labels**, use the backbone network as a feature extractor and do the rest as described above.

3. Repeating the same with G-SimCLR (backbone ResNet34) model with and without multi-crop augmentation on a small sample of PV dataset (Apple class, 10 healthy, 10 unhealthy).

4. Repeating the same with SwAV over various set of small samples (Apple class 20 healthy vs 10 unhealthy, 5 healthy vs 5 unhealthy etc.).

Before delving into the results of these experiments, it is worth to explain the underlying concept behind G-SimCLR, Self-Supervised Contrastive Learning with Guided Projection via Pseudo Labelling [207]. As mentioned before, SimCLR aims to maximize the similarity between the two resulting representations for a given image while also minimize the similarity between the very same image and then resulting representations from the other images. It's quite probable that there might be near-duplicate or similar images already appearing in the same dataset and SimCLR would still try to treat even the near-duplicate images as negative pairs of one another, causing high loss that slows down the convergence. Even if the negative impact of this phenomenon is slightly prevented by large batches and large amount of unlabelled images, it would still be undesirable when we have small amount of images.

In order to attack this, G-SimCLR proposes a method for generating information about the label space without using explicit labels, and then using this information to improve the quality of representations learned through SimCLR. This approach involves training a denoising autoencoder in a self-supervised manner by minimizing the reconstruction loss between the input and noisy images, and using the latent space representations from the trained autoencoder to cluster the input space of images and assign each image to a corresponding cluster, referred to as pseudo

labels. In simple terms, the image representations from encoder are clustered and the batches are created by picking images from different (dissimilar) clusters. These pseudo labels are then incorporated into the determination of training batches for SimCLR to reduce the risk of images from the same label space being included in the same batch. The steps applied in G-SImCLR can be seen at Figure 5.9 and the cluster visualizations (with tSNE) can be seen at Figure 5.10.

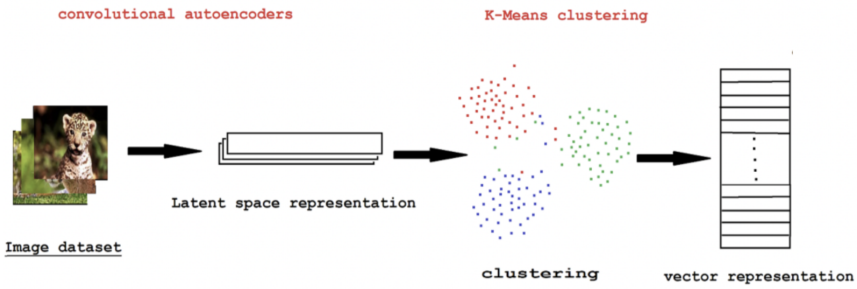


Figure 5.9: In G-SimCLR, the latent space representations of all the images in the dataset is clustered and then the batches to feed in SimCLR training process are generated via these clusters to reduce the risk of images from the same label space being included in the same batch.

The results from these experiments are depicted in Figure 5.11. These results demonstrate that the G-SimCLR method with multi-crop augmentation achieves the highest accuracy (0.9113) among all the experiments; even higher than what we get from the approach that we collect the feature embeddings through the backbone of fully supervised trained network (0.8767). This indicates that SSL would be a strong player for dealing with small datasets where the labelled samples are scarce or expensive to gather.

5.3 Salient Image Segmentation as a Surprising Player in SSL

5.3.1 Introduction: Salient Image Segmentation

Salient image segmentation is a CV task that involves identifying the most prominent or eye-catching objects or regions in an image. It plays a crucial role in

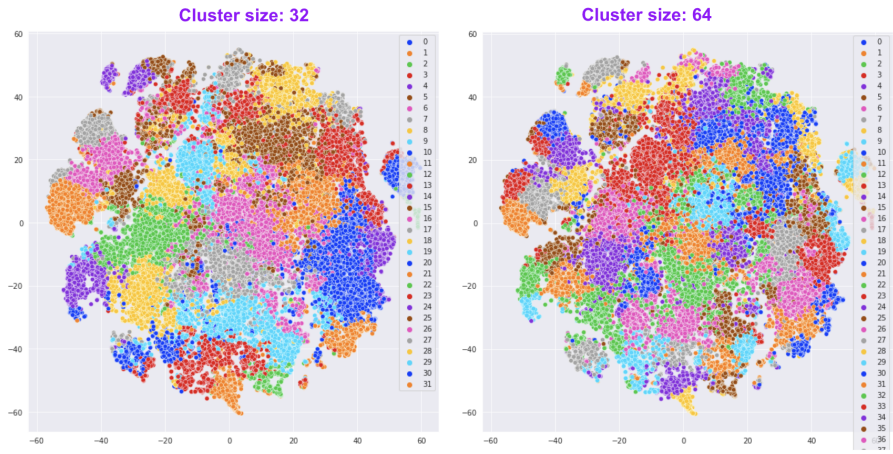


Figure 5.10: The visualizations of clusters that is generated by the latent space representations (vectors) of all the images in the PV dataset show that there are several distinct clusters that we can leverage to form the batches accordingly.

various image and video analysis applications such as object recognition, scene understanding, and image retrieval. The goal of salient image segmentation is to automatically highlight the most important regions in an image, which may include objects, backgrounds, or even textures. These regions are typically characterized by their visual distinctiveness, such as their color, shape, size, or texture.

There are several approaches to salient image segmentation, including bottom-up approaches that rely on low-level image features and top-down approaches that leverage high-level context and prior knowledge. In recent years, there has been a significant amount of research on developing deep learning-based approaches for salient image segmentation, which have shown promising results on various benchmark datasets [208]. Despite the progress made in the field, salient image segmentation remains a challenging task due to the variability and complexity of visual scenes. It is also a highly dynamic task, as the saliency of an image can change depending on the context and the viewer's focus of attention.

We propose an augmentation policy for Contrastive SSL in the form of an already established Salient Image Segmentation technique entitled Global Contrast based Salient Region Detection. This detection technique, which had been devised for

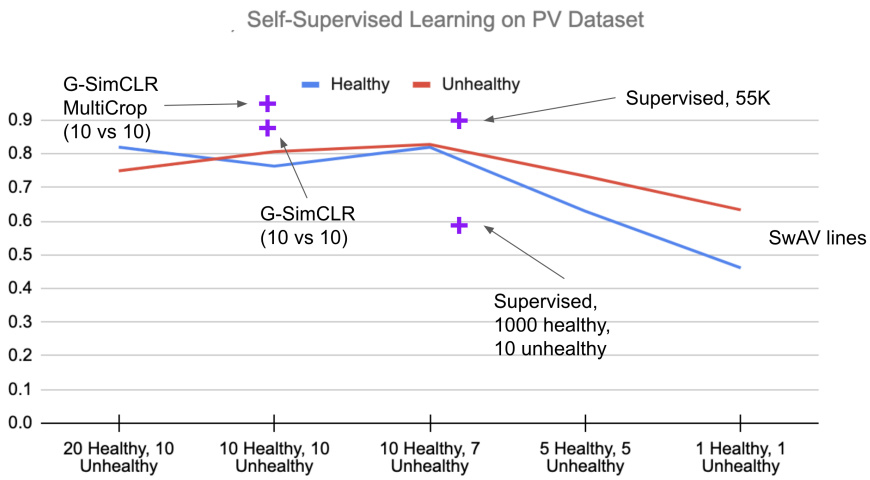


Figure 5.11: The chart above shows the performance of various approaches for classifying healthy and unhealthy plant leaves using different numbers of labeled and unlabeled samples. The x-axis represents the number of samples from both healthy and unhealthy leaves in the test set, while the y-axis indicates the classification accuracy achieved by each method. The performance of each approach was evaluated using a 10-epoch SSL training process on the entire Plant Village (PV) dataset (size 55K) without any labels, followed by a logistic regression classification for 50-epochs using reduced samples. The results show that G-SimCLR with multicrop augmentation and only 10 labeled samples from each class performed the best, outperforming both a fully supervised version using the entire 55K labeled dataset and a supervised training approach with 1000 healthy samples and 10 unhealthy samples. In addition, SwAV was also found to perform better than the supervised approach, but not better than G-SimCLR.

unrelated Computer Vision tasks, was empirically observed to play the role of an augmentation facilitator within the SSL protocol. This observation is rooted in our practical attempts to learn, by SSL-fashion, aerial imagery of solar panels, which exhibit challenging boundary patterns. Upon the successful integration of this technique on our problem domain, we formulated a generalized procedure and conducted a comprehensive, systematic performance assessment with various Contrastive SSL algorithms subject to standard augmentation techniques. This evaluation, which was conducted across multiple datasets, indicated that the proposed technique indeed contributes to SSL. We hypothesize whether salient image segmentation may suffice as the only augmentation policy in Contrastive SSL when treating downstream segmentation tasks.

Despite recent advances in CV, the effectiveness of visual recognition and learning still very much depends on manual annotations and on how well they represent the images at hand. Given the substantial amount of available unlabeled data and the costly manual annotation, new methods for online and unsupervised learning are crucial for achieving robust and efficient visual learning. Current unsupervised learning methods do not perform learning under a genuine unsupervised state – they rather focus on transfer learning, or unsupervised tasks subject to strong features that were pre-trained in a supervised way. Therefore, there is a need to investigate algorithms for unsupervised learning of completely new categories, such as SSL to learn in a continuous, long-term fashion. Indeed, SSL needs also to generalize the classical unsupervised learning scenario of physical objects to scenarios of higher-level actions and more complex activities, and at the same time develop capabilities to detect abnormalities in visual data [209], [210].

The requirement for very large datasets of manually labeled instances may seem counter-intuitive since this is not how humans learn to recognize new objects. Humans are constantly fed with images through their eyes, and are able to learn an object's appearance and to distinguish it from other objects without knowing what the object exactly is. Moreover, collecting large-scale datasets is time-consuming and expensive, while the supervised approach to learn features from labeled data has almost reached its saturation due to the intense labor required in manually annotating millions of data instances. This is because most of the modern CV systems (that are supervised) try to learn some form of image representations by finding a pattern that links the data points to their respective annotations in large datasets [187]. Moreover, the data annotation efforts vary from task to task, and it is estimated that the time spent on image segmentation and object detection (i.e., carefully drawing boundaries) is four times longer than the image classification itself [211]. The annotation efforts become significantly higher when it comes to highly regulated and specialized domains like medicine and finance in which the expertise level of the human annotator matters more than in any other domain. Moreover, supervised learning not only depends on expensive annotations but also suffers from other drawbacks such as generalization errors, spurious correlations, and being prone to adversarial attacks [31].

In this section, we explore the viability of using an unsupervised image segmentation

technique called 'Global Contrast based Salient Region Detection (SGD)' [100] as an augmentation policy (rather than a proxy task) in contrastive SSL methods and study its impact on downstream supervised image segmentation tasks in low data regimes. To the best of our knowledge, this is the first study that explores salient object segmentation techniques as an augmentation policy in contrastive SSL.

We will show that using this SGD algorithm as an image augmentation policy in SSL produces better representations for downstream image segmentation tasks when compared to default augmentation policies commonly utilized in SSL methods. In order to make the integration of SGD into SSL pretraining routines feasible, we also devise a simple manipulation called offline augmentation with hashing that enables running comprehensive experiments with various parameters and configurations. We will also provide evidence that SGD-based augmentation policy in SSL performs better with low resolution images.

The current section targets the following *research questions*:

> Would SGD produce better representation when used as an augmentation policy in Contrastive SSL? Moreover, would salient image segmentation suffice as the only augmentation policy in Contrastive SSL when treating downstream segmentation tasks?

The concrete contributions of this section are the following:

- Comparing a relatively old unsupervised image segmentation technique, Global Contrast based Salient Region Detection (SGD), with recent deep learning (DL)-based image segmentation algorithms.
- Evaluating the viability of a generative model (Pix2Pix) as a proxy for computationally expensive image augmentation methods.
- Devising an SGD-based efficient offline augmentation technique to incorporate any expensive augmentation policy in DL training routines.
- Employing SGD as an offline augmentation policy (rather than a proxy task) in contrastive SSL methods and study its impact on downstream supervised image segmentation and object detection tasks.

- Illustrating that fine-grained details in high resolution images would negatively impact the performance of SSL when compared to the coarse-grained details in low resolution images.

- Formulating a recommendation for choosing the most appropriate SSL method (accounting for the the augmentation technique, downstream task and even the imagery resolution).

The remainder of the section is organized as follows: Section 5.3.2 describes the concrete motivation that ignited this research, and then summarizes related work as well as various SSL approaches, including the role of data augmentation therein. It concludes with presenting the SGD method in detail. Section 5.3.4 outlines the experimental setup regarding SGD, including the datasets the various preliminary efforts to make SGD-based augmentation policies viable in DL training routines. It then presents the attained results. Section 5.3.5 discusses the findings and proposes possible *mechanistic* explanations, and finally Section 5.3.6 concludes by pointing out the key points and future directions.

5.3.2 Background

Explicitly, SSL is a machine learning process where the model trains itself to learn one part of the input from another part of the input. In other words, the model learns from labels that are presumably already intrinsic in the data itself. This process is also known as predictive or pretext learning and the unsupervised problem is transformed into a supervised problem by auto-generating the labels. To effectively exploit the huge quantity of unlabeled data, it is crucial to set the right learning objectives, in order to obtain the appropriate supervision from the data itself.

Transfer learning (TL) has been presented as an effective solution for constructing robust feature representations when the training set for a given problem is small. As its name suggests, TL aims to transfer knowledge and learned features from one task (the source task) to another related target task, just as a person can utilize the same knowledge across different projects. To do this, TL trains the model on a large labeled dataset and then treats this model as a starting point in the target task's training, without learning from scratch. This dataset creates the

target task's representation model, using the same architecture as the source task. The initialized representation network in the target task is then further trained on the target dataset. So we can say that the workflows of SSL and TL are similar, with only slight differences. The key difference between TL and SSL is that TL pre-trains on labeled data, whereas SSL utilizes unlabeled data to learn features and need only a small number of labelled example [212].

Within the context of utilizing unlabeled data to learn the underlying representations, SSL can be organized as (i) handcrafted pretext tasks-based, (ii) contrastive learning-based and (iii) clustering learning-based approaches. In handcrafted pretext tasks-based method, a popular approach has been to propose various pretext tasks that help in learning features using pseudo-labels while the networks can be trained by learning objective functions of the pretext tasks and the features are learned through this process [187]. Tasks such as image-inpainting [213], colorizing gray-scale images [214], solving jigsaw puzzles [215], image super-resolution [216], video frame prediction [217], audio-visual correspondence [218], to mention the most prominent, have proven to be effective for learning good representations. In doing so, the model learns quality representations of the samples and is used later for transferring knowledge to downstream tasks. The selection of an appropriate proxy task (pretext) is critical to the effectiveness of self-supervised learning. It requires careful design, and indeed, numerous researchers investigated various approaches for given downstream tasks. For example, [219] proposed a proxy task to classify whether a given pair of kidneys belong to the same side; with the assumption that the network needs to develop an understanding of the structure and sizes of the kidneys.

On the other hand, contrastive learning (CL) is a training method wherein a classifier distinguishes between "similar" (positive) and "dissimilar" (negative) input pairs. It is essentially the task of grouping similar samples closer to each other, unlike setting diverse samples far from each other. During training, the augmented version of the original sample is considered as a positive sample, and the rest of the samples in the batch/dataset (depends on the method being used) are considered negative samples. Next, the model is trained in a way that it learns to differentiate positive samples from the negative ones. Constructing positive and negative pairs via data augmentation allows the model to learn from inter-class variance

(uniformity) by pushing negative pairs far away, and from intra-class similarity (alignment) by pulling positive pairs together. The core concept is to maximize the dot product between the feature vectors which are similar and minimize the dot product between those of which are not similar. CL methods use contrastive loss that is evaluated based on the feature representations of the images extracted from an encoder network. As briefly explained in section 5.1.3, the popular methods that recently started to produce results comparable to the state-of-the-art supervised learning methods, even with less labelled images, can be regarded as DINO [202], SwAV [200], MoCo [198], [203] , BYOL [196], SimSiam [199] and SimCLR, [220], [195]. The representation extraction strategies differ from one method to another (e.g. BYOL does not even need negative pairs) but the changes are very subtle and without rigorous ablations, it is hard to tell which one works better on a case at hand. The widely used approach to evaluate the learned representations through the SSL's pre-training process is the linear evaluation protocol [221], where a linear classifier (e.g., SVM, Logistic Regression, etc.) is trained on top of the frozen backbone network (image representations are derived from the final or penultimate layers of the backbone network as a feature vector). Finally, the test accuracy is used as a proxy for representation quality (Figure 5.12).

Data augmentation is critical for effective learning in contrastive SSL methods, and the composition of multiple data augmentation operations is key to defining effective contrastive prediction tasks that yield high-quality representations. Stronger data augmentation is particularly beneficial for contrastive SSL when compared to supervised learning, with techniques such as color distortion and cropping playing a key role in producing effective views for the considered dataset [195]. These findings underscore the importance of carefully selecting and combining data augmentation techniques when developing contrastive SSL models for a wide range of applications.

Augmentations can be regarded as an indirect way to pass human prior knowledge into the model, and an effective augmentation should discard the unimportant features for the downstream task (i.e., removing the "noise" to classify an image). The nature of the selected augmentations should fit the downstream task, maintain the image semantics (meaning), introduce the model with challenging assignments/tests. Their selection depends upon the dataset's underlying distribution and cardinality, whereas a recent study [222] further claims that augmentations

are problem-class-dependent.

Let us add more clarity to crafting an augmentation policy that should give the model a hard time and why its selection is dependent on the dataset's diversity and size. In contrastive SSL, we feed pairs/different views of the same image to the network and these views should be visually dissimilar in a way that the model should have a hard time while bringing them together if the views are coming from the same image or pushing them apart if they are coming from different images. If a weak augmentation is applied, it will be an easy job for the network to puss/pull; hence the learning will not be strong enough. Dataset's diversity and size play a role here as the more diverse the dataset, the easier it gets for the model to push dissimilar views (in low diversity datasets in which there many similar images, it is harder to push dissimilar views).

The most popular data augmentation techniques in contrastive SSL include rotation, crop, cut, flip, color jitter, blur, Gaussian noise and gray scale. Almost all of the contrastive SSL methods use these (or similar) augmentation techniques at certain degrees no matter what the downstream task is. Selection of the most suitable augmentation policy is a crucial step in contrastive SSL. [195] systematically studied the impact of data augmentation and observed that no single transformation suffices to learn good representations, even though the model can almost perfectly identify the positive pairs in the contrastive task. When composing augmentations, the contrastive prediction task becomes harder, but the quality of representation improves dramatically. Compared to handcrafted pretext tasks-based methods, contrastive SSL methods are easier to implement but the pretraining process is computationally expensive as they require large batches and large amount of unlabelled datasets. Nevertheless, the pretrained backbone ConvNets through contrastive SSL methods generally produce better latent representations and perform well on downstream tasks.

Importantly, even though classical image segmentation methods can be regarded as one of the pretext tasks in SSL, the downstream segmentation and object detection tasks usually try to detect/segment certain salient objects and areas in an image, rather than entire textures, which unsupervised segmentation algorithms generate. While essentially solving a segmentation problem, salient object detection and

segmentation approaches segment only the salient foreground object from the background, rather than partitioning an image into regions of coherent properties as in general segmentation algorithms. That is, using a salient image segmentation in SSL is likely to generate better representations when the downstream task is defined as either image segmentation or object detection.

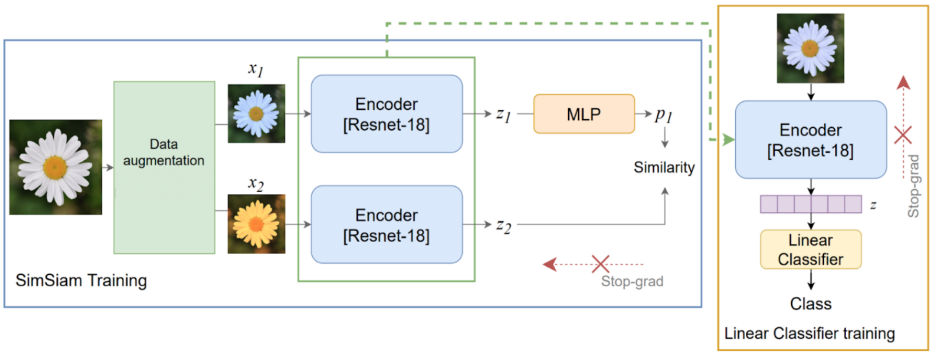


Figure 5.12: Linear evaluation protocol to evaluate the learned representations of a pretraining process of SimSiam [199].

5.3.2.1 Global Contrast based Salient Region Detection (SGD)

SGD, also known as SaliencyCut [100], is an automated unsupervised salient region extraction method, an improved iterative version of GrabCut [223], which introduced a contrast analysis method to integrate spatial relationships into region-level contrast computation. In GrabCut, the user initially draws a rectangle around the foreground region in an image, and then the algorithm iteratively segments it to get the best result. The executed steps are (i) estimating the color distribution of the foreground and background via a Gaussian Mixture Model (GMM), (ii) constructing a Markov random field over the pixels labels (i.e., foreground vs. background), and (iii) applying a graph cut optimization to arrive at the final segmentation. Instead of manually selecting this rectangular region to initialize the process, SaliencyCut is using histogram-based contrast (HC) and region-based contrast (RC) techniques (that are also suggested in the same paper) to create saliency maps and then binarizes this map using a fixed threshold (e.g., value of 70).

In RC, the input image is first segmented into regions, then the color contrast at the region level is computed, and saliency for each region is finally defined as the weighted sum of the region's contrasts to all other regions in the image. The weights are set according to the spatial distances with farther regions being assigned smaller weights.

In HC, the number of colors needed to consider is reduced to 1728 by quantizing each color channel to have 12 different values (12 color values per channel in R-G-B). Considering that color in a natural image typically covers only a small portion of the full color space, the number of colors is further reduced by ignoring less frequently occurring colors. By choosing more frequently occurring colors, and by ensuring that they cover the colors used *de facto* by more than 95% of the image pixels, we are typically left with around n = 85 colors. The colors of the remaining pixels, which comprise fewer than 5% of the image pixels, are replaced by the closest colors in the histogram.

Once the saliency map is initialized with RC and HC, GrabCut is iteratively run (i.e., iterative refinements) to improve the SaliencyCut result. After each iteration, dilation and erosion operations are used on the current segmentation result to get a new trimap for the next GrabCut iteration. During this iterative refinement, adaptive fitting is used (regions closer to an initial salient object region are more likely to be part of that salient object than far-away regions). Thus, the new initialization enables GrabCut to include nearby salient regions, and exclude non-salient regions according to color dissimilarity. See [100] for more details regarding SGD and Figure 5.13.

Finally, Figure 5.14 presents the outcomes' comparison of SGD-based segmentation versus latest unsupervised segmentation on PASCAL VOC 2012 [225].

5.3.3 Related Work

Semantic segmentation is the task of assigning each pixel to a specific class label. The class labels can be the same as for object detection, but unlike the object detection task, which labels each instance of an object as separate objects, semantic segmentation only assigns a pixel a specific class label and does not differentiate instances of objects.

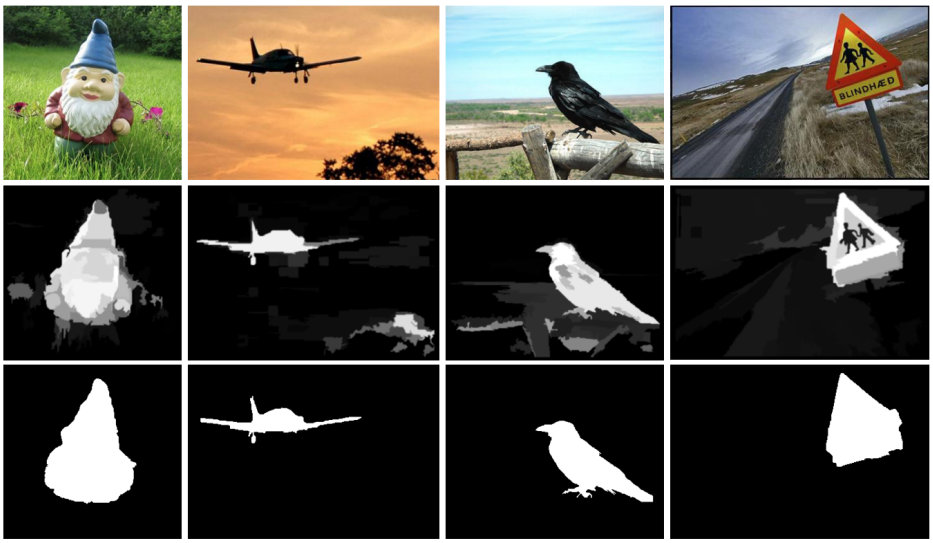


Figure 5.13: Given an input image (top), a global contrast analysis is used to compute a saliency map (middle), which can be used to produce an unsupervised segmentation mask (bottom) for an object of interest [224].

As the augmentations in any SSL method should be tailored *a priori* and fit in terms of the downstream tasks, devising an augmentation policy for downstream segmentation tasks depends on harnessing the intrinsic features that help model learn how to segment the objects in an image. This is usually achieved through auxilary tasks such as rotating, cropping, colorization etc. One of the most useful auxilary tasks can be regarded as image colorization that is introduced in [214] as a process of estimating RGB colors for grayscale images. The backbone network within the pretrained model performed well for downstream tasks like object classification, detection, and segmentation compared to other methods. The study in [229] also suggested a similar technique to automatically colorize grayscale images by merging local information dependent on small image patches with global priors computed using the entire image. Since these two techniques employ a strategy of finding suitable reference images and transferring their color onto a target grayscale image, the semantic information plays a little role and has the potential to actually hamper the SSL. Probably one of the most useful colorization tasks is suggested by [230], in which a system must interpret the

semantic composition of the scene (what is in the image) as well as to localize objects (where things are) to incorporate semantic parsing and localization into a colorization system. In a generative manner, [213] proposed image inpainting, a context-based pixel prediction where the network understands the context of the entire image as well as the hypothesis for missing parts, hence assisting in extracting the semantic information.

There are many other auxiliary tasks proposed in SSL but most of these methods focus on basic inherent visual features, like image patches and rotation, which are very simple tasks that are not likely to completely learn the semantics and the spatial features of the image. Actually, the default augmentation policies in contrastive SSL methods are the derivation of these auxiliary tasks. To the best of our knowledge, salient image segmentation has not been used as an auxiliary task in any SSL method before, let alone contrastive SSL.

5.3.4 Implementation, Setup & Results

Given the aforementioned research question, we derive concrete experimentation tasks:

1. Preliminary: Applying SGD to the PVs' datasets
2. Obtaining a solid and an efficient implementation
3. Testing the primary hypothesis: SGD as an augmentation policy

5.3.4.1 Setup and Datasets

We used the following datasets in the current study:

- **Aerial Drone Images for Solar Photovoltaic (PV) Panels Defects (NORCE-PV):** This dataset is provided by NORCE (Norwegian Research Centre) and has 790 high resolution aerial drone images. It is originally gathered for PV panels defect detection classification and annotated by NORCE internally. In this study, we will not be dealing with defect classes but the images themselves to run the experiments. The dataset is not publicly available.

- **Multi-resolution PV Dataset From Satellite and Aerial Imagery (MultiRes-PV):** This dataset includes three groups of PV samples collected at the spatial resolution of 0.8m, 0.3m and 0.1m, namely PV08 from Gaofen-2 and Beijing-2 imagery, PV03 from aerial photography, and PV01 from UAV orthophotos. In this study, we only used PV01 rooftop images (645 in total) that comes with segmentation masks that come within 256x256 size [231].

- **CIFAR10:** The CIFAR-10 dataset consists of 60,000 32x32 colour images in 10 classes, with 6000 images per class. There are 50,000 training images and 10,000 test images [232].

- **STL10:** The STL-10 dataset is a subset of ImageNet, consists of 1300 labelled, 100,000 96x96 unlabelled colour images in 10 classes. In particular, each class has fewer labeled training examples than in CIFAR-10, but a very large set of unlabeled examples is provided to learn image models prior to supervised training [233].

5.3.4.2 Preliminary: Running SGD on NORCE-PV and MultiRes-PV Datasets

SGD is a computationally expensive process and it produces segmentations on various level of details given the size of an image. For instance, it takes $\sim$ 2min to generate a saliency map (segmentation) of an 600x450 RGB image, while it takes around $\sim$ 1sec to accomplish the same process on 60x45 (10% of the original image). In the cases wherein a highly detailed segmentation map is needed, SGD implementation would be practically impossible to implement on real time. In some other cases wherein a rough segmentation map would suffice, applying SGD on a reduced size of an image would still produce useful segmentation. On the other hand, when we apply SGD on low resolution images like MultiRes-PV images (256x256), we may need to upscale the image to get a better segmentation. The outcome of applying SGD on various sizes of NORCE-PV (high resolution), NORCE-PV (high resolution, zoomed-in) and MultiRes-PV (low resolution) images is presented in Figures 5.16, Figures 5.17 and 5.18, respectively.

We also run STEGO [234] and DETIC [102], two recent popular zero-shot DL-based image segmentation algorithms, on our datasets and compare results with SGD. As

evident in Figure 5.15, DETIC obtains impressive results as it is already capable of detecting solar arrays from natural images (it is trained to detect 20 thousand different objects out of the box). Compared to the heavy STEGO algorithm, developed just a few months ago, SGD performs comparatively better.

5.3.4.3 An Efficient Implementation

Employing SGD on any DL training routine as an augmentation or transformation policy is highly inefficient in terms of computational costs. Notably, using GPU is not an option due to a layered (a mixture of OpenCV and Python functions) image preprocessing techniques along with RC and HC methods. The fact that thousands of images are to pass through this process in each epoch renders the integration of SGD in the DL training process practically infeasible. Our tests indicate that training a Resnet-18 ConvNet with SGD augmentation to classify PV defects using the NORCE-PV dataset (790 images, shape 32x32), for one epoch, takes $\sim$ 50 min on Tesla K80 (11.5GB) GPU.

Image to Segmentation Map Translation via Pix2Pix To achieve feasiblity, we firstly thought about training a generative DL model to learn SGD segmentations via image translation and then replace the SGD process with this generative model. We chose the Pix2Pix architecture [235] for this purpose and trained several models. We basically fed the original images and SGD-segmented version to the network and trained further to get the model learn translating one image to another. Using this Pix2Pix model within the Resnet-18 training routine to replace SGD reduced the duration from $\sim$ 50 min to $\sim$ 30 min per epoch. Even though the results look promising (Figure 5.19), there are several drawbacks, e.g., information loss during the translation process, being outperformed on image instances that were under-represented, and the limited speed-improvement gains – which altogether render this process unviable in practice.

Offline Augmentation with Hashing SGD is a computationally expensive method that results in a speed bottleneck when used in DL training routines and it might be one of the reasons that such a strong segmentation method could not establish itself despite all the latest advancements in similar fields. Since SGD is basically a deterministic approach, the segmentation map is always the same except colors; hence the contours and overall shapes are always identical for an

image. In other words, even if the pixel colors change at each iteration of SGD, the overall shapes and lines in an image are always the same, unless resizing is applied. Experimenting around this attribute, we devised an unprecedented solution by running SGD once for all the images in the dataset and creating a segmentation mask for each image and then reusing that every epoch during training without running SGD again and again. In order to simulate the random colors and make the model invariant to colors (so it can focus more on other features, e.g., contrast, shapes, lines etc.), we also applied random color jittering and swapping from the same color palette utilized by SGD. Next, we describe the explicit steps:

(Step-1) Read every image in the dataset as a `numpy` array and hash it to store the entire image as a single hash code (string),

(Step-2) Run SGD over every image in the dataset and save the segmentation map as an image to the disk,

(Step-3) Create a dictionary (key-value pairs) with the hash strings (of Step-1) as keys and the file path of every segmentation map as the associated value (if reading from disk becomes a bottleneck, all of the segmentation maps can be read at once and stored in the memory as an array),

(Step-4) In model training with original images, get the hashmap of every image array and find the file path of corresponding segmentation map from the dictionary above at each iteration, and read that image from the disk or from memory if it is stored in memory,

(Step-5) Apply color jittering to randomly swap the colors of the augmented image and use that as a proxy augmented image during the rest of the process.

This technique allowed SGD process to run instantly (practically below 1sec) as the segmentation maps are read from disk/memory at real-time at no cost.

5.3.4.4 Using SGD as an Augmentation Policy in Contrastive SSL Algorithms

In the Contrastive SSL pre-training routine, the augmented version of the original sample is considered as a positive sample, and the rest of the samples in the

batch/dataset (depends on the method being used) are considered negative samples. Then, the model is trained in a way that it learns to differentiate positive samples from the negative ones. In doing so, the model learns quality representations of the samples and is used later for transferring knowledge to downstream tasks.

A predefined CV architecture (e.g., ResNet50, VGG19, EfficientNet etc.) is typically used as a ConvNet backbone, and the weights (model parameters) are updated during this contrastive SSL process. Then, the backbone is saved and used in another architecture as a starting point or just as a facilitator of feature extraction (representations from one of the last fully connected (FC) layers).

In this study, in order to quickly iterate across various heavy augmentation combinations under limited computational resources, we used a lightweight ResNet18 architecture and then pretrained several backbones with the combination of SGD versus standard image augmentation techniques (crop, grayscaling, jittering etc.) using the following SSL algorithms: SimSiam [199], BYOL [196], SimCLR [195], MoCo [198], SwAV [200] and Barlow Twins [201]. Then we tested the effectiveness of these backbones in downstream image clustering, image segmentation, object detection and classification problems.

Image Clustering Using SSL Backbones Using the 80% of the entire NORCE-PV dataset with no label, we trained SvAW, SimSiam and SimCLR models for 100 epochs with the combinations of default SSL augmentations and SGD (with offline augmentation through hashing as well as Pix2Pix versions), used their backbone networks to extract the image representations (vectors) of 20% of the dataset, and then used KMeans and t-SNE methods to cluster and visualize the clusters. The outcome is presented in Figure 5.20. Evidently, SGD, when pretrained on unlabelled images, enables better image representations (vectors, embeddings) in all of the tested cases. The more distant the clusters, the better the representations generated by the SSL backbone network. In fact, observing this clear delineation between the clusters on a 2D space had originally given us an idea of applying SGD as an augmentation on larger scales with other combinations on a downstream image segmentation task.

Image Segmentation and Object Detection Using SSL Backbones As a downstream image segmentation architecture, we used Detectron2 [236] framework

on MultiRes-PV dataset to detect and segment solar panels from rooftop aerial images. In order to pretrain the SSL algorithms, the following augmentation policies are applied (p within the parenthesis corresponds to the probability of applying the respective technique, and sample augmentations are shown in Figure 5.21). The clear difference between random augmentation versus SGD augmentation can also be seen in Figure 5.22).

- Default SSL augmentations (*Random Resized Crop, Random Rotate, Random Horizontal Flip, Random Vertical Flip, Random Color Jittering, Random Grayscale, Gaussian Blur at various probabilities,* p)
- SGD ($p = 1.0$),
- SGD ($p = 1.0$) with default SSL augmentations
- SGD ($p = 0.5$) with default SSL augmentations
- SGD ($p = 1.0$) with jittering ($p = 1.0$)
- SGD ($p = 0.5$) with jittering ($p = 1.0$)
- SGD ($p = 0.8$) with jittering ($p = 0.8$)

The batch size is known to be one of the most important parameters in SSL and the larger the batch size, the better the representations as most of SSL algorithms greatly benefit from large batches (in order to have larger number of negatives available in mini batches) but we could not investigate performance for more than 128 images in a batch due to limited computational resources and the high number of experiments that we need to run (more than 500 runs, each takes at least a few hours). In order to test the impact of image resizing and batch sizing on SGD in image segmentation tasks, we picked three parameter combinations with five SSL methods (SimSiam, SimCLR, BYOL, MoCo, Barlow Twins): (i) Offline SGD applied on the original size ($100p$) with batch size 16 ($bs16$), (ii) Offline SGD applied on the original size ($100p$) with batch size 128, (iii) Offline SGD applied on the upscaled size ($200p$) with batch size 16.

Upscaling (x2) is tested to see the impact of detailed SGD segmentations, and it is found that the performance reaches its peak when SGD is applied on the original image-size with a batch size of 128. We also tested with doubling the resolution of each image and then applying SGD but the impact of SGD on SSL performance was not as good as in the original image-size. This can be explained by the fact

that the color variation in SGD with low resolution (original size) images is more or less similar to what we can expect to see in a solar panel segmentation. Once we increase the resolution by up-scaling, the details on a solar panel become more evident and the SGD may assign different colors for such areas, which we do not want to see for a simple panel segmentation task (e.g., every object is represented by a single color in image segmentation).

Then, using 500 images as a training set (out of 645 images from MultiRes-PV dataset), we pretrained all of the five SSL methods in a training loop with image segmentation and object detection tasks and measured the test accuracy on the test set (145 images). We observed that performances of SSL methods vary given the augmentation policy, as shown in Figure 5.23. For instance, while MoCo performs better compared to other SSL methods, with default SSL augmentations yet no SGD, SimSiam and BYOL perform better compared to other SSL methods with SGD plus default SSL augmentations as well as with only default augmentations. In another case, SimCLR performs the best when SGD is applied at $p = 0.5$ along with default augmentations. This result tells us that each SSL method performs differently, as a function of the augmentation policy, while the impact of SGD also varies under different settings.

Then, we experimented with the same SSL methods, having additionally no-SSL (with default ImageNet checkpoints) at various levels of labeled images (subsets from 10% to 100%) and measured the test accuracy on the same 145 images. For instance, we picked at first 50 labeled images from the training set, then using a pretrained SSL backbone from the previous step, we trained an image segmentation and object detection model with Detectron2, and tested the models on the test set. Then we did the same for 20%, 30% and so on. The AP metrics for the selected SSL methods for selected augmentation are shown in Figure 5.24. As seen from this figure, in every level of the reduced training set, an augmentation policy with an SGD component usually performs better than an SSL with default augmentations as well as a vanilla (no-SSL) model training. In some cases, using only SGD augmentation would even suffice without further application of augmentation.

Image Classification Using SSL Backbones As a downstream image classification model, using CIFAR10 and STL10 datasets, we used Logistic Regression

to classify the image representations taken from the penultimate layers of SimCLR backbones that are pretrained on CIFAR10 training set (5000 unlabelled images) with the augmentation policies mentioned before (with hashing applied to SGD augmentations). During these experiments, various levels of labeled images from CIFAR10 (from 10% to 100%) are picked at each iteration and only one SSL method (SimCLR) is picked for the sake of brevity. The results from this experiment can be seen in Figure 5.25). Even though there is a slim performance gain (around 5% improvement) with SGD plus default augmentations applied, the default pretrained ImageNet backbone still performs better without SSL. This can be explained by the smaller number of utilized epochs (100) while pretraining with SimCLR and the large number of labelled images being used in the ImageNet pretraining. Considering that SimCLR is pretrained only by 5000 unlabelled images from CIFAR10 while the default ResNet18 model is pretrained with the entire ImageNet dataset with ground truth labels, getting similar metrics with SimCLR is still worth mentioning herein.

We run a similar experiment on the STL10 dataset (using 100 thousand unlabeled images), but SGD did not contribute in that case and performed worse. Since STL is already a subset of the ImageNet dataset, the default pretrained ImageNet backbone did quite well and exceed all the metrics achieved with SSL methods. The results are shown in Figure 5.26. Given the fact that the images in the STL10 dataset have higher resolution (96x96) than the images in CIFAR10 (32x32), getting worse performance with SSL using SGD can be explained by the same phenomenon we observed in SGD of upscaled images doing worse compared to original size images.

5.3.5 Discussion

Our experiments with clustering the image representations via SGD-augmented backbone networks indicate that SGD enables obtaining better image representations in all of the cases that we tested. Even if we may not know the number of clusters within a dataset, the clear delineation between the clusters indicates a better representation to separate one image from another using visual clues. Usually, the most salient details and objects play the role of separators between one image from another, but using classical image augmentation policies (e.g., cropping,

jittering, etc.) totally fails on this end. At the same time, SGD-based salient object segmentation does a better job on catching those critical details, thus assisting more in downstream clustering tasks.

We argue that using a salient image segmentation in SSL generates better representations when the downstream task is image segmentation or object detection due to the fact that salient object detection and segmentation approaches segment only the salient foreground object from the background, rather than partition an image into regions of coherent properties as in general segmentation algorithms. Our experiments with various augmentation combinations with and without SGD show that the augmentation policy having an SGD component usually performs better than an SSL with default augmentations as well as a vanilla (no SSL) model training. In some cases, using only SGD augmentation would even suffice, with no further application of augmentation.

Another important observation is that each SSL method performs differently as a function of the underlying augmentation policy, whereas the impact of SGD also varies under different settings. For instance, while MoCo performs better compared to other SSL methods, with default SSL augmentations yet no SGD, SimSiam and BYOL perform better compared to other SSL methods with SGD plus default SSL augmentations as well as with only default augmentations, as was elaborated in the previous section. Notably, this in fact the common question in the age of DL in which there are a myriad of alternatives: How to know which algorithm does better on a certain use case and how to pick the right one? Not only the SSL method but also the augmentation technique, downstream task and even the resolution of your images matter.

One of the most unexpected observation of our tests can be regarded as having worse results with SGD applied on high resolution images. We have observed this effect on various occasions during our experiments as reported in the previous section. This can be explained by the fact that the color variation in SGD with low resolution (original size) images is lower and more similar to what we can expect to see in a coarse grained segmentation tasks. Once we increase the resolution by up-scaling, the details on an image becomes more evident and SGD may assign different colors for such areas, which we do not want to see for a simple image

segmentation task (e.g., every object is represented by a single color in image segmentation). However, SSL performing well on low resolution images can also be attributed to the low number of epochs (i.e., 100) as the model is not able to catch the low level details in high resolution images in such a limited number of epochs.

Nevertheless, a recent SSL study also supports our claim and sheds some light on this issue as follows: [237] claims that current self-supervised methods learn representations that can easily disambiguate coarse-grained visual concepts like those in ImageNet. However, as the granularity of the concepts becomes finer, self-supervised performance lags further behind supervised baselines.

5.3.6 Conclusions

By empirically addressing the posed research questions, our investigation confirmed the capacity of SGD to play a role in modern SSL. Overall, this study successfully leveraged SGD, a 10-years-old salient object detection and segmentation algorithm, as a potent image augmentation technique for downstream image segmentation tasks. To achieve an effective integration of SGD into SSL pretraining routines, we devised a simple manipulation called offline augmentation with hashing. This fine implementation detail enabled us to run hundreds of SSL experiments with various parameters and configurations. We then demonstrated that using a salient image segmentation in SSL generates better representations when the downstream task is image segmentation. Our experiments with clustering the image representations via SGD-augmented backbone networks indicate that SGD helps better image representations in all of the cases tested. Our experiments with various augmentation policies including SGD show that the augmentation policy having a SGD component usually does better than a SSL with default augmentations; in some cases, using only SGD augmentation alone would even be be better.

Our experiments with MultiRes-PV, CIFAR10 and STL10 also showed that SSL with SGD-based augmentation policy performs well with low resolution images. This still remains to be verified whether fine-grained features and/or the low number of epochs played a role in this observation. We contend that salient object segmentation algorithms produce coarse grained segmentation due to saliency, thus perform well on segmenting coarse grained objects like PV solar panels and it may

not hold true if our goal were to extract fine grained details in an image.

We observed that each SSL method performs differently given the augmentation policy, whereas the impact of SGD also varies under different settings. In our opinion, the most unexpected observation of our tests can be regarded as having worse results with SGD applied to high resolution images compared to low resolution ones. We conclude that the augmentation technique, type of a downstream task and image resolution are the most important elements that have a high impact on the success of a SSL method picked.

Figure 5.14: Comparing the outcomes of unsupervised versus SGD-based segmentation on PASCAL VOC 2012. Invariant Information Clustering (IIC) [226], Superpixels [227], Continuity Loss [228]. Different segments are shown in different colors (the images in the first 5 rows are taken from [228]). Notably, SGD was developed at least 8 years prior to IIC and Superpixels, and can still perform comparatively better on some cases.

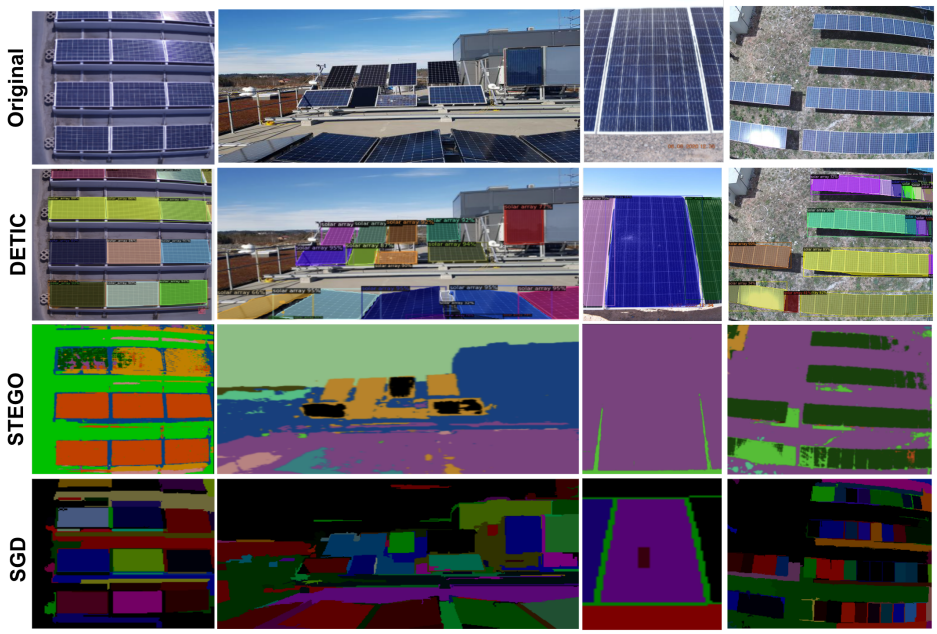


Figure 5.15: Comparison of transformer-based zero-shot segmentation methods versus SGD on NORCE PV dataset.

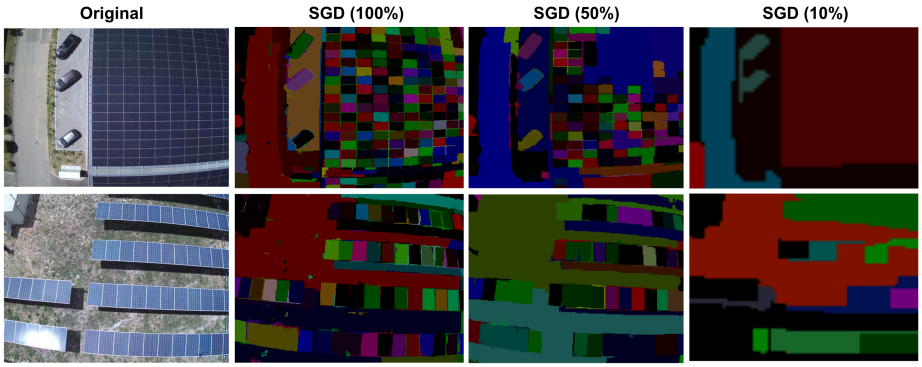


Figure 5.16: Segmentation produced by SGD from NORCE-PV images with various sizes.

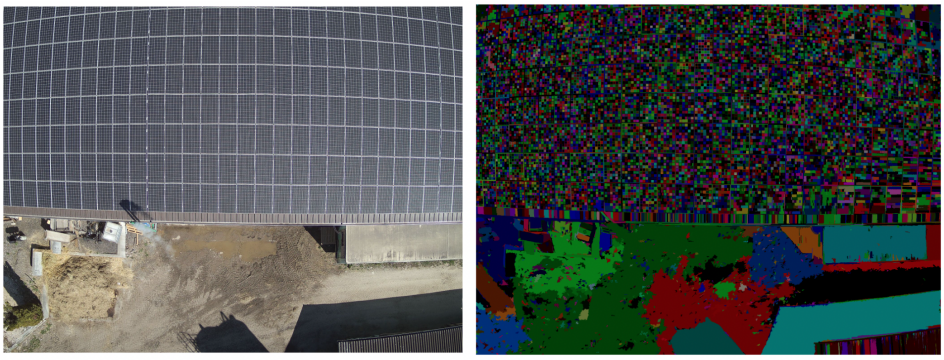


Figure 5.17: Segmentation produced by SGD from a NORCE-PV image with original resolution (dimension 4000x3000, resolution 72x72). The higher the resolution, the better the segmentation at fine grained detailed.

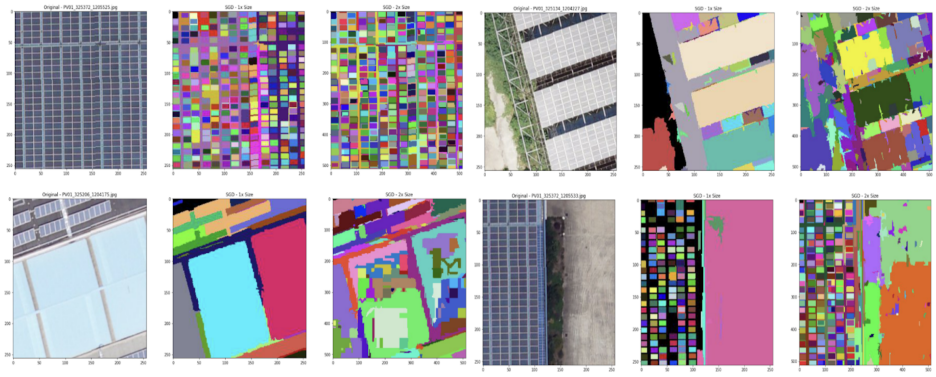


Figure 5.18: Segmentation produced by SGD from low resolution MultiRes-PV images with original versus 200% rescaled.

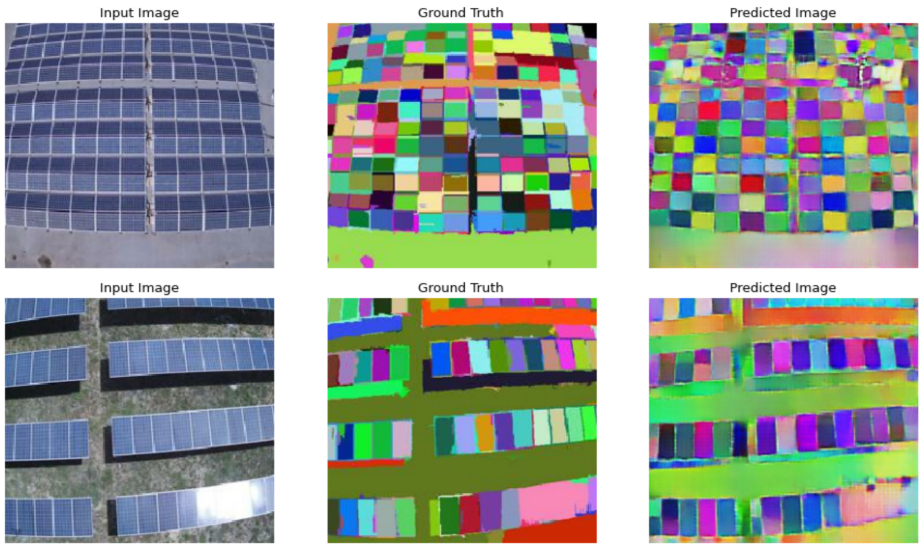


Figure 5.19: The second column of this figure shows the SGD segmentation of the original images and the images generated with Pix2Pix image translation model are shown at the third column.

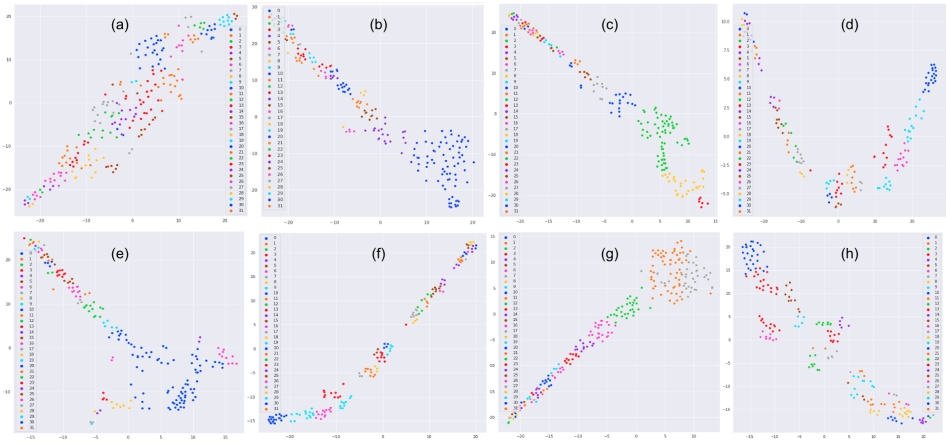


Figure 5.20: Clustering partial NORCE-PV dataset image representations produced by (a) ResNet18 ImageNet weights (b) SimSiam random aug. (c) SimSiam random + PixPix based SGD aug. (d) SimCLR random + SGD aug. (e) SimSiam random + SGD aug. f) SimSiam with PixPix based SGD aug. (g) SwAV with SGD aug. (h) SimSiam with SGD aug.

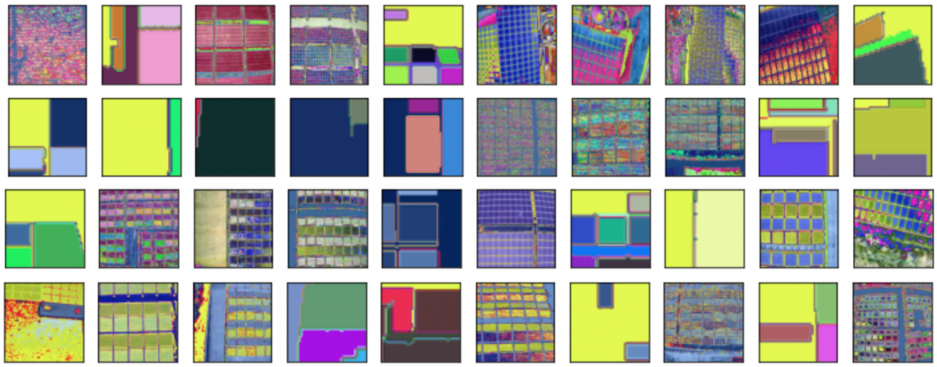


Figure 5.21: This figure shows a batch of 40 NORCE-PV images. Every image in this batch is segmented by SGD at first and then the other default random augmentations are applied. Basically, this is what the model sees at each iteration.

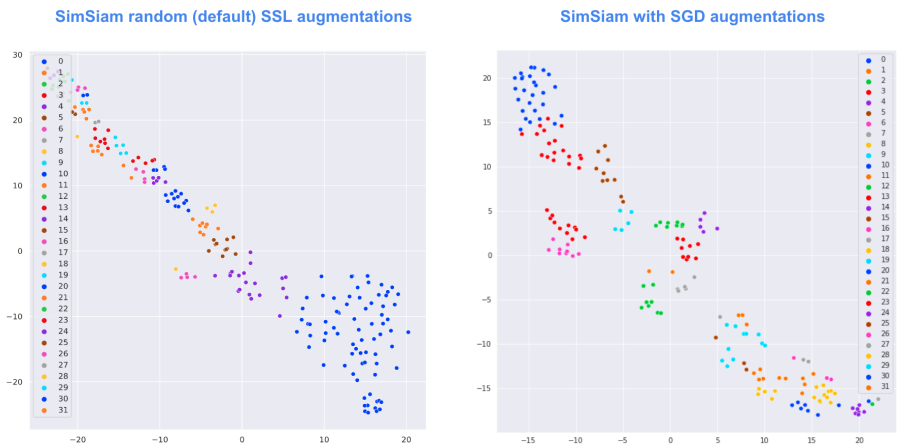


Figure 5.22: Clustering partial NORCE-PV dataset image representations produced by ResNet18 ImageNet weights finetuned with SimSiam using random augmentation versus SGD augmentation that returns much better delineation between the clusters representing different classes.

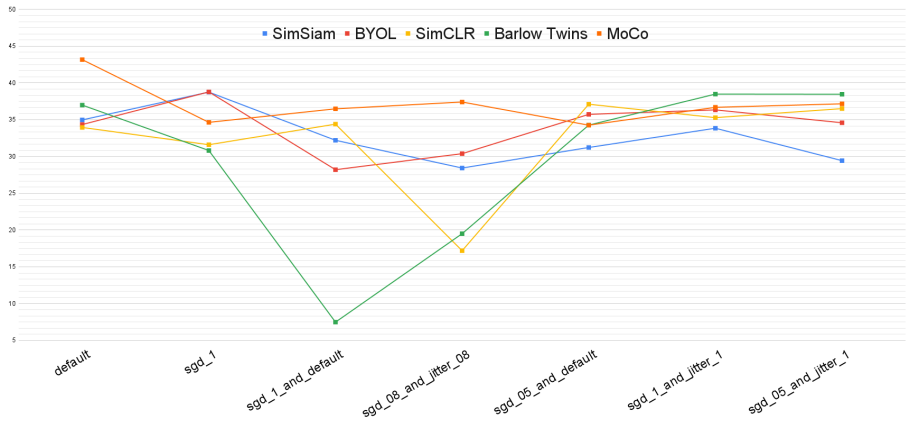


Figure 5.23: Selecting the best combination of SGD and default augmentation strategies for various SSL methods using original size images with batch size 128. The performance of augmentation strategies varies by the SSL methods accompanied. SGD alone without any other augmentation strategies produced the best result in BYOL and SimSiam compared to other methods and augmentation policies.

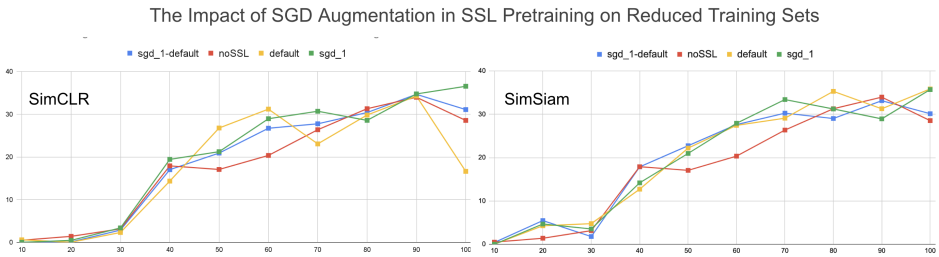


Figure 5.24: Segmentation metrics for SGD with SimSiam and SimCLR experiments with reduced number of images from the training set at various levels. x-axis denotes the percentage of the samples with respect to entire training set (from 10% to 100%) and y-axis denotes test set average precision (AP) of segmentation model trained with the partial training set.

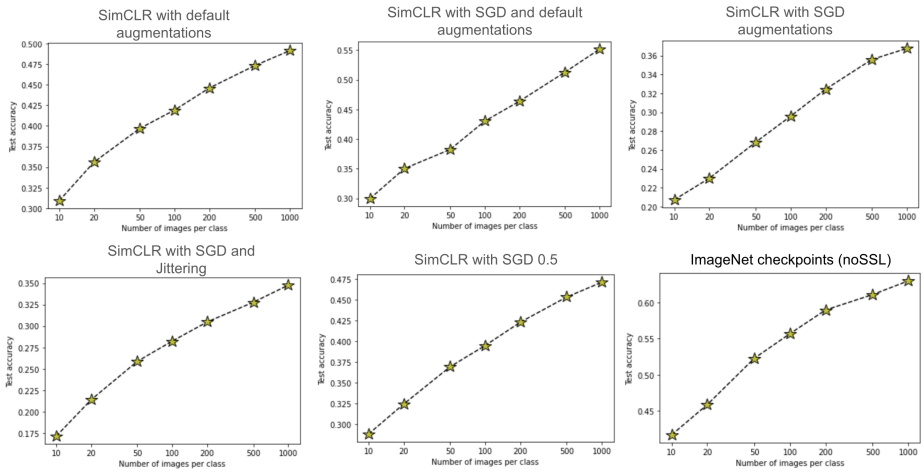


Figure 5.25: Classification of images in CIFAR10 dataset using partial samples from training set and SSL method with SGD applied. SGD plus default augmentations outperforms default SSL augmentations by 5% and a little worse than ImageNet checkpoints.

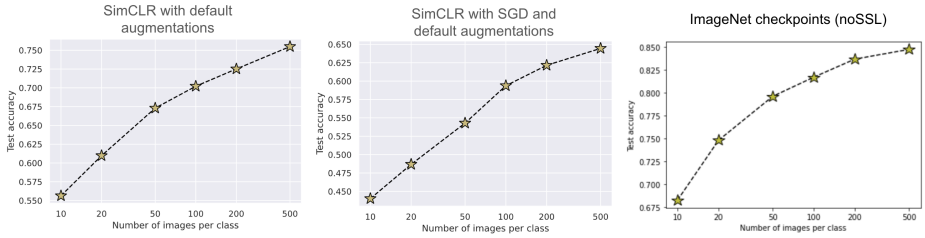


Figure 5.26: Classification of images in STL10 dataset using partial samples from training set and SSL method with SGD applied. SGD did worse than default SSL augmentations and the default pretrained ImageNet backbone outperforms all the metrics achieved with SSL methods (due to the fact that STL is already a subset of ImageNet dataset).

Chapter 6

Conclusions

In conclusion, this thesis has delved into the topic of learning from small samples in machine learning and provided a comprehensive overview of the various approaches that have been proposed to effectively tackle this problem.

The third chapter presented an overview of the various techniques used in overcoming small data problems, including data selection and preprocessing, incorporating prior knowledge, ensemble methods, transfer learning, regularization techniques, and synthetic data generation, among others.

The approaches discussed in this chapter covered a wide range of methods, including data selection and preprocessing, incorporation of domain, prior and context knowledge, ensemble methods, transfer learning, parameter initialization, loss function reformulation, regularization techniques, data augmentation, synthetic data generation, problem reduction, optimization techniques, using physics-informed neural networks, unsupervised learning techniques, semi-supervised learning, self-supervised learning, zero-shot, one-shot and few-shot learning, meta learning, harnessing model uncertainty, active learning, self-learning, multi-task learning, symbolic learning, hierarchical learning, knowledge distillation based learning, and dealing with imbalanced data.

The forth chapter focused on the impact of batch normalization in learning from small samples, particularly in the context of imbalanced datasets. To simulate such scenarios, we artificially generate skewness (99% vs. 1%) for certain plant types out of the PlantVillage dataset as a basis for classification of scarce visual cues through transfer learning. By randomly and unevenly picking healthy and unhealthy samples from certain plant types to form a training set, we consider a base experiment as fine-tuning ResNet34 and VGG19 architectures and then testing the model performance on a balanced dataset of healthy and unhealthy images. We empirically observe that the initial F1 test score jumps from 0.29 to 0.95 for the minority class upon adding a final Batch Normalization (BN) layer just before the output layer in VGG19.

In order to find if there is a certain ratio in which the impact is maximized, we experimented with various levels of imbalance ratios and conditions, and observed that the impact of final BN on highly imbalanced settings is the most obvious when the ratio of minority class to the majority is less than 10%; above that almost no impact. As expected, the impact of the final BN layer is more obvious on minority class than it is on majority class, albeit the level of impact with respect to the imbalance ratio is almost same, and levels off around 10%.

We also experimented if the batch size would also be an important parameter for the minority class test accuracy when the final BN is added and found out that the highest score is gained when the batch size is around 64, whereas the accuracy drops afterwards with larger batches. Additionally, we empirically demonstrated that the final BN layer could still be eliminated in inference without compromising the attained performance gain.

Our calibration experiments show that a network has much lower ECE, i.e. has a more ideal confidence relative to its own accuracy. In effect, a final BN-layered network is not 'over-confident' and as a result generalizes and performs better on unseen datasets.

As a result, we empirically illustrated that adding a batch normalization layer before the softmax output layer significantly reduced the training time and improved the test error for minority classes, resulting in a more than three-fold performance

boost in some configurations.

The fifth chapter explored the role of self-supervised learning as an augmentation policy in learning from small samples. The study showed that using salient image segmentation in self-supervised learning improved the representations learned, especially in the context of the downstream task of image segmentation.

More specifically, this chapter investigated the impact of the Global Contrast based Salient Region Detection (SGD) algorithm on self-supervised learning. I demonstrated the potential of SGD as a powerful image augmentation technique for image segmentation tasks in self-supervised learning. The implementation of SGD into SSL pretraining routines was achieved through a simple manipulation called offline augmentation with hashing. The experiments carried out showed that using SGD as an augmentation policy in SSL generates better representations for image segmentation tasks.

The results also indicated that SSL with SGD-based augmentation performed well with low resolution images, but this needs further investigation. The results showed that different SSL methods perform differently based on the augmentation policy used and that the impact of SGD also varies in different settings. An unexpected observation was the worse results obtained with SGD applied to high-resolution images compared to low-resolution ones. The results of this study highlighted the importance of the augmentation technique, type of downstream task, and image resolution in determining the success of a SSL method. These findings can be used to guide future studies on self-supervised learning and the application of SGD in this field.

One promising avenue for further investigation in the area of self-supervised learning pretraining would be the integration of offline augmentation with hashing, which could yield valuable insights into the performance of unsupervised and zero-shot segmentation algorithms, as well as salient object detection techniques. Additionally, our results highlight the potential benefits of exploring other types of augmentation policies in the pretraining process, and we encourage researchers to continue exploring these and other approaches to improve the performance and versatility of self-supervised learning models in a wide range of applications.

In summary, this thesis highlights the importance of effectively learning from small samples in machine learning and the various approaches that can be used to tackle this problem. The results of this study demonstrate the potential of batch normalization and self-supervised learning in improving the performance of models trained on small datasets. These findings have important implications for researchers and practitioners working in the field of machine learning, and open up avenues for further exploration and innovation in the area of learning from small samples.

Chapter 7

Bibliography

[1] Veysel Kocaman, Ofer M Shir, and Thomas Bäck. Improving model accuracy for imbalanced image classification tasks by adding a final batch normalization layer: An empirical study. In *2020 25th International Conference on Pattern Recognition (ICPR)*, number 10.1109/ICPR48806.2021.9412907, pages 10404–10411. IEEE, 2021.

[2] Veysel Kocaman, Ofer M Shir, and Thomas Bäck. The unreasonable effectiveness of the final batch normalization layer. In *International Symposium on Visual Computing*, volume 13018, pages 81–93. Springer, 2021.

[3] Veysel Kocaman, Ofer M Shir, Thomas Bäck, and Ahmed Nabil Belbachir. Saliency can be all you need in contrastive self-supervised learning. In *International Symposium on Visual Computing*, pages 119–140. Springer, 2022.

[4] Veysel Kocaman and David Talby. Biomedical named entity recognition at scale. In *International Conference on Pattern Recognition*, pages 635–646. Springer, Cham, 2021.

[5] Veysel Kocaman and David Talby. Improving clinical document understanding on covid-19 research with spark nlp. *AAAI-21 Workshop on Scientific*

Document Understanding, (arXiv preprint arXiv:2012.04005), 2020.

[6] Veysel Kocaman and David Talby. Spark nlp: natural language understanding at scale. *Software Impacts*, 8:100058, 2021.

[7] Veysel Kocaman and David Talby. Accurate clinical and biomedical named entity recognition at scale. *Software Impacts*, 13:100373, 2022.

[8] Veysel Kocaman, Bunyamin Polat, Gursev Pirge, and David Talby. Biomedical named entity recognition in eight languages with zero code changes. In *Proceedings of the Iberian Languages Evaluation Forum (IberLEF 2022)*, volume 3202, 2022.

[9] Veysel Kocaman, Youssef Mellah, Hasham Haq, and David Talby. Automated de-identification of arabic medical records. In *Proceedings of ArabicNLP 2023*, pages 33–40, 2023.

[10] Veysel Kocaman, Hasham Ul Haq, and David Talby. Beyond accuracy: Automated de-identification of large real-world clinical text datasets. In *Machine Learning for Health (ML4H) 2023–Findings track*, 2023.

[11] Veysel Kocaman, Hasham Ul Haq, and David Talby. Beyond accuracy: Automated de-identification of large real-world clinical text datasets. In *Proceedings of The Professional Society for Health Economics and Outcomes Research (ISPOR) Europe 2023*, Copenhagen, Denmark, 2023. Poster presentation.

[12] Sutanay Choudhury, Khushbu Agarwal, Colby Ham, Pritam Mukherjee, Siyi Tang, Sindhu Tipirneni, Veysel Kocaman, Suzanne Tamang, Robert Rallo, and Chandan K Reddy. Tracking the evolution of covid-19 via temporal comorbidity analysis from multi-modal data. In *AMIA*, 2021.

[13] Syed Raza Bashir, Shaina Raza, Veysel Kocaman, and Urooj Qamar. Clinical application of detecting covid-19 risks: A natural language processing approach. *Viruses*, 14(12):2761, 2022.

[14] Khushbu Agarwal, Sutanay Choudhury, Sindhu Tipirneni, Pritam Mukherjee,

Colby Ham, Suzanne Tamang, Matthew Baker, Siyi Tang, Veysel Kocaman, Olivier Gevaert, et al. Preparing for the next pandemic via transfer learning from existing diseases with hierarchical multi-modal bert: a study on covid-19 outcome prediction. *Scientific Reports*, 12(1):1–13, 2022.

[15] Hasham Ul Hak, Veysel Kocaman, and David Talby. Deeper clinical document understanding using relation extraction. In *https://arxiv.org/abs/2112.13259*. Scientific Document Understanding workshop at AAAI 2022, 2021.

[16] Hasham Ul Hak, Veysel Kocaman, and David Talby. Mining adverse drug reactions from unstructured mediums at scale. In *W3PHIAI workshop at AAAI-22*. https://arxiv.org/abs/2201.01405, 2022.

[17] Murat Aydogan and Veysel Kocaman. Trsav1: A new benchmark dataset for classifying user reviews on turkish e-commerce websites. *Journal of Information Science*, 1:https–journals, 2022.

[18] Vikas Kumar, Lawrence Rasouliyan, Veysel Kocaman, and David Talby. Using natural language processing to identify adverse drug events of angiotensin converting enzyme inhibitors. International Forum on Quality and Safety in Healthcare EUROPE 2021, 2021.

[19] A. Emre Varol, Veysel Kocaman, Hasham Ul Hak, and David Talby. Understanding covid-19 news coverage using medical nlp. In *5th International Workshop on Narrative Extraction from Texts (Text2Story)*, 2022.

[20] Hasham Ul Haq, Veysel Kocaman, and David Talby. Connecting the dots in clinical document understanding with relation extraction at scale. *Software Impacts*, 12(100294), 2022.

[21] Vikas Kumar, Lawrence Rasouliyan, Veysel Kocaman, and David Talby. Detecting adverse drug events in dermatology through natural language processing of physician notes. In *36th International Conference on Pharmacoepidemiology & Therapeutic Risk Management*, volume 36, pages https–www, 2022.

[22] Juan Martinez, Veysel Kocaman, Hasham Ul Haq, and David Talby. Zero-shot

information extraction for clinical nlp. [under review].

[23] Arshaan Nazir, Thadaka Kalyan Chakravarthy, David Amore Cecchini, Rakshit Khajuria, Prikshit Sharma, Ali Tarik Mirik, David Talby, and Veysel Kocaman. Langtest: A comprehensive evaluation library for custom llm and nlp models. [under review].

[24] Julio Bonis, Veysel Kocaman, and David Talby. Social determinants of health in clinical narratives: A comprehensive review of pubmed clinical case reports from 1975 to 2022. Available at SSRN: https://ssrn.com/abstract=4590921 or http://dx.doi.org/10.2139/ssrn.4590921, 2022.

[25] Yuanyuan Pu, Derek B Apel, Victor Liu, and Hani Mitri. Machine learning methods for rockburst prediction-state-of-the-art review. *International Journal of Mining Science and Technology*, 29(4):565–570, 2019.

[26] Daniel T Chang. Exemplar-based contrastive self-supervised learning with few-shot class incremental learning. *arXiv preprint arXiv:2202.02601*, 2022.

[27] David Haussler and Manfred Warmuth. The probably approximately correct (pac) and other learning models. *The Mathematics of Generalization*, pages 17–36, 2018.

[28] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[29] Alvaro Figueira and Bruno Vaz. Survey on synthetic data generation, evaluation methods and gans. *Mathematics*, 10(15):2733, 2022.

[30] Lisa Torrey and Jude Shavlik. Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, pages 242–264. IGI global, 2010.

[31] Xiao Liu, Fanjin Zhang, Zhenyu Hou, Li Mian, Zhaoyu Wang, Jing Zhang, and Jie Tang. Self-supervised learning: Generative or contrastive. *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[32] Xu Sun, Weiwei Sun, Shuming Ma, Xuancheng Ren, Yi Zhang, Wenjie Li, and Houfeng Wang. Complex structure leads to overfitting: A structure regularization decoding method for natural language processing. *arXiv preprint arXiv:1711.10331*, 2017.

[33] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021.

[34] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *ICLR*. OpenReview.net, 2019.

[35] Thomas G Dietterich. Ensemble methods in machine learning. In *Multiple Classifier Systems: First International Workshop, MCS 2000 Cagliari, Italy, June 21–23, 2000 Proceedings 1*, pages 1–15. Springer, 2000.

[36] Tian Jin, Michael Carbin, Dan Roy, Jonathan Frankle, and Gintare Karolina Dziugaite. Pruning's effect on generalization through the lens of training and regularization. *Advances in Neural Information Processing Systems*, 35:37947–37961, 2022.

[37] Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. Grokking: Generalization beyond overfitting on small algorithmic datasets. *arXiv preprint arXiv:2201.02177*, 2022.

[38] Najeeb Khan, Jawad Shah, and Ian Stavness. Bridgeout: stochastic bridge regularization for deep neural networks. *IEEE Access*, 6:42961–42970, 2018.

[39] Amulya Agarwal and Nitin Gupta. Comparison of outlier detection techniques for structured data. *arXiv preprint arXiv:2106.08779*, 2021.

[40] CJ Adcock. Sample size determination: a review. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 46(2):261–283, 1997.

[41] Russell V Lenth. Some practical guidelines for effective sample size determination. *The American Statistician*, 55(3):187–193, 2001.

[42] Rosa L Figueroa, Qing Zeng-Treitler, Sasikiran Kandula, and Long H Ngo. Predicting sample size required for classification performance. *BMC medical informatics and decision making*, 12:1–10, 2012.

[43] Myunggwon Hwang, Yuna Jeong, and Wonkyung Sung. Data distribution search to select core-set for machine learning. In *The 9th International Conference on Smart Media and Applications*, pages 172–176, 2020.

[44] Mansheej Paul, Surya Ganguli, and Gintare Karolina Dziugaite. Deep learning on a data diet: Finding important examples early in training. *Advances in Neural Information Processing Systems*, 34:20596–20607, 2021.

[45] Ben Goertzel. Artificial general intelligence: concept, state of the art, and future prospects. *Journal of Artificial General Intelligence*, 5(1):1, 2014.

[46] Alexander Ratner, Stephen H Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. Snorkel: Rapid training data creation with weak supervision. In *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases*, volume 11, page 269. NIH Public Access, 2017.

[47] Salva Rühling Cachay, Benedikt Boecking, and Artur Dubrawski. End-to-end weak supervision. *Advances in Neural Information Processing Systems*, 34:1845–1857, 2021.

[48] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[49] Tirtharaj Dash, Sharad Chitlangia, Aditya Ahuja, and Ashwin Srinivasan. A review of some techniques for inclusion of domain-knowledge into deep neural networks. *Scientific Reports*, 12(1):1–15, 2022.

[50] Ziyang Jiang, Tongshu Zheng, and David Carlson. Incorporating prior knowledge into neural networks through an implicit composite kernel. *arXiv preprint arXiv:2205.07384*, 2022.

[51] Mohsen Behzad, Keyvan Asghari, Morteza Eazi, and Maziar Palhang. Gen-

eralization performance of support vector machines and neural networks in runoff modeling. *Expert Systems with applications*, 36(4):7624–7629, 2009.

[52] Yarin Gal. *Uncertainty in deep learning.* PhD thesis, 2016.

[53] Matthew Olson, Abraham Wyner, and Richard Berk. Modern neural networks generalize on small data sets. *Advances in Neural Information Processing Systems*, 31, 2018.

[54] Shuo Feng, Huiyu Zhou, and Hongbiao Dong. Using deep neural network with small dataset to predict material defects. *Materials & Design*, 162:300–310, 2019.

[55] Giorgio Corani and Marco Zaffalon. Learning reliable classifiers from small or incomplete data sets: The naive credal classifier 2. *Journal of Machine Learning Research*, 9(4), 2008.

[56] Dioanjan Sarkar and Raghav Bali. *Transfer Learning in Action.* Manning publication, 2021.

[57] Deepak Soekhoe, Peter van der Putten, and Aske Plaat. On the impact of data set size in transfer learning using deep neural networks. In *International symposium on intelligent data analysis*, pages 50–60. Springer, 2016.

[58] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *Advances in neural information processing systems*, 27, 2014.

[59] Thorsten Hoeser and Claudia Kuenzer. Object detection and image segmentation with deep learning on earth observation data: A review-part i: Evolution and recent trends. *Remote Sensing*, 12(10):1667, 2020.

[60] Simon Kornblith, Jonathon Shlens, and Quoc V Le. Do better imagenet models transfer better? In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2661–2671, 2019.

[61] Andrew B Sellergren, Christina Chen, Zaid Nabulsi, Yuanzhen Li, Aaron

Maschinot, Aaron Sarna, Jenny Huang, Charles Lau, Sreenivasa Raju Kalidindi, Mozziyar Etemadi, et al. Simplified transfer learning for chest radiography models using less data. *Radiology*, 305(2):454–465, 2022.

[62] Gabriel Michau and Olga Fink. Unsupervised transfer learning for anomaly detection: Application to complementary operating condition transfer. *Knowledge-Based Systems*, 216:106816, 2021.

[63] Andrew Arnold, Ramesh Nallapati, and William W Cohen. A comparative study of methods for transductive transfer learning. In *Seventh IEEE international conference on data mining workshops (ICDMW 2007)*, pages 77–82. IEEE, 2007.

[64] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

[65] Andre Perunicic. Understanding neural network weight initialization. *URL https://intoli. com/blog/neural-network-initialization*, 2017.

[66] Bjorn Barz and Joachim Denzler. Deep learning on small datasets without pretraining using cosine loss. In *The IEEE Winter Conference on Applications of Computer Vision*, pages 1371–1380, 2020.

[67] John T Hancock and Taghi M Khoshgoftaar. Survey on categorical data for neural networks. *Journal of Big Data*, 7(1):1–41, 2020.

[68] Lorenzo Rosasco, Ernesto De Vito, Andrea Caponnetto, Michele Piana, and Alessandro Verri. Are loss functions all the same? *Neural computation*, 16(5):1063–1076, 2004.

[69] Katarzyna Janocha and Wojciech Marian Czarnecki. On loss functions for deep neural networks in classification. *arXiv preprint arXiv:1702.05659*, 2017.

[70] Qishan He, Lingjun Zhao, Gangyao Kuang, and Li Liu. Sar target recognition based on model transfer and hinge loss with limited data. In *CAAI*

International Conference on Artificial Intelligence, pages 191–201. Springer, 2021.

[71] N Srivastavanitish, Geoffrey Hinton, A Krizhevskykriz, I Sutskeverilya, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res*, 15(1):1929–1958, 2014.

[72] Jonathan Tompson, Ross Goroshin, Arjun Jain, Yann LeCun, and Christoph Bregler. Efficient object localization using convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 648–656, 2015.

[73] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[74] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015.

[75] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48, 2019.

[76] Alhassan Mumuni and Fuseini Mumuni. Data augmentation: A comprehensive survey of modern approaches. *Array*, page 100258, 2022.

[77] Antreas Antoniou, Amos Storkey, and Harrison Edwards. Data augmentation generative adversarial networks. *arXiv preprint arXiv:1711.04340*, 2017.

[78] Hiroshi Inoue. Data augmentation by pairing samples for images classification. *arXiv preprint arXiv:1801.02929*, 2018.

[79] Abdul Jabbar, Xi Li, and Bourahla Omar. A survey on generative adversarial networks: Variants, applications, and training. *ACM Computing Surveys (CSUR)*, 54(8):1–49, 2021.

[80] Alexander Jung. imgaug. `https://github.com/aleju/imgaug`, 2020.

[81] Yu Yang, Lei Sun, Xiuqing Mao, and Min Zhao. Data augmentation based on

generative adversarial network with mixed attention mechanism. *Electronics*, 11(11):1718, 2022.

[82] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 113–123, 2019.

[83] Samuel G Müller and Frank Hutter. Trivialaugment: Tuning-free yet state-of-the-art data augmentation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 774–782, 2021.

[84] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[85] Haibo He, Yang Bai, Edwardo A Garcia, and Shutao Li. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)*, pages 1322–1328. IEEE, 2008.

[86] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[87] Shengguo Hu, Yanfeng Liang, Lintao Ma, and Ying He. Msmote: Improving classification performance when training data is imbalanced. In *2009 second international workshop on computer science and engineering*, volume 2, pages 13–17. IEEE, 2009.

[88] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

[89] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[90] Veit Sandfort, Ke Yan, Perry J Pickhardt, and Ronald M Summers. Data augmentation using generative adversarial networks (cyclegan) to improve generalizability in ct segmentation tasks. *Scientific reports*, 9(1):16884, 2019.

[91] Fabio Henrique Kiyoiti dos Santos Tanaka and Claus Aranha. Data augmentation using gans. *arXiv preprint arXiv:1904.09135*, 2019.

[92] Georgios Douzas and Fernando Bacao. Effective data generation for imbalanced learning using conditional generative adversarial networks. *Expert Systems with applications*, 91:464–471, 2018.

[93] Sajila D Wickramaratne and Md Shaad Mahmud. Conditional-gan based data augmentation for deep learning task classifier improvement using fnirs data. *Frontiers in big Data*, 4:659146, 2021.

[94] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[95] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 5907–5915, 2017.

[96] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

[97] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.

[98] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[99] Venkatesh Boddapati, Andrej Petef, Jim Rasmusson, and Lars Lundberg.

Classifying environmental sounds using image recognition networks. *Procedia computer science*, 112:2048–2056, 2017.

[100] Mingming CHENG and Guoxin ZHANG. Niloy j, et al. global contrast based salient region detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Colorado Spring, USA*, pages 409–416, 2011.

[101] Iago Suárez, José M Buenaposada, and Luis Baumela. Elsed: Enhanced line segment drawing. *arXiv preprint arXiv:2108.03144*, 2021.

[102] Xingyi Zhou, Rohit Girdhar, Armand Joulin, Philipp Krähenbühl, and Ishan Misra. Detecting twenty-thousand classes using image-level supervision. In *European Conference on Computer Vision*, pages 350–368. Springer, 2022.

[103] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *International conference on learning representations*, 2016.

[104] Xiaohua Zhai, Avital Oliver, Alexander Kolesnikov, and Lucas Beyer. S4l: Self-supervised semi-supervised learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1476–1485, 2019.

[105] Yongjun Xu, Xin Liu, Xin Cao, Changping Huang, Enke Liu, Sen Qian, Xingchen Liu, Yanjun Wu, Fengliang Dong, Cheng-Wei Qiu, et al. Artificial intelligence: A powerful paradigm for scientific research. *The Innovation*, 2(4):100179, 2021.

[106] Nathan Baker, Frank Alexander, Timo Bremer, Aric Hagberg, Yannis Kevrekidis, Habib Najm, Manish Parashar, Abani Patra, James Sethian, Stefan Wild, et al. Workshop report on basic research needs for scientific machine learning: Core technologies for artificial intelligence. Technical report, USDOE Office of Science (SC), Washington, DC (United States), 2019.

[107] Kejun Tang, Xiaoliang Wan, and Chao Yang. Das-pinns: A deep adaptive sampling method for solving high-dimensional partial differential equations. *Journal of Computational Physics*, 476:111868, 2023.

[108] Ben Moseley, Andrew Markham, and Tarje Nissen-Meyer. Finite basis physics-informed neural networks (fbpinns): a scalable domain decomposition approach for solving differential equations. *Advances in Computational Mathematics*, 49(4):62, 2023.

[109] Ben Moseley, Andrew Markham, and Tarje Nissen-Meyer. Solving the wave equation with physics-informed deep learning. *arXiv preprint arXiv:2006.11894*, 2020.

[110] Ben Moseley. harmonic-oscillator-pinn. `https://github.com/benmoseley/harmonic-oscillator-pinn`, 2022.

[111] YH Taguchi. *Unsupervised feature extraction applied to bioinformatics: A PCA based and TD based approach*. Springer Nature, 2019.

[112] Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, 20(3):542–542, 2009.

[113] Yassine Ouali, Céline Hudelot, and Myriam Tami. An overview of deep semi-supervised learning. *arXiv preprint arXiv:2006.05278*, 2020.

[114] Yue Fan, Anna Kukleva, Dengxin Dai, and Bernt Schiele. Revisiting consistency regularization for semi-supervised learning. *International Journal of Computer Vision*, pages 1–18, 2022.

[115] Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning. *arXiv preprint arXiv:1610.02242*, 2016.

[116] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. *Advances in neural information processing systems*, 30, 2017.

[117] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1979–1993, 2018.

[118] Yves Grandvalet and Yoshua Bengio. Semi-supervised learning by entropy minimization. *Advances in neural information processing systems*, 17, 2004.

[119] Dong-Hyun Lee et al. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3, page 896, 2013.

[120] Fan Ma, Deyu Meng, Xuanyi Dong, and Yi Yang. Self-paced multi-view co-training. *Journal of Machine Learning Research*, 2020.

[121] Avital Oliver, Augustus Odena, Colin A Raffel, Ekin Dogus Cubuk, and Ian Goodfellow. Realistic evaluation of deep semi-supervised learning algorithms. *Advances in neural information processing systems*, 31, 2018.

[122] Yi Xu, Jiandong Ding, Lu Zhang, and Shuigeng Zhou. Dp-ssl: Towards robust semi-supervised learning with a few labeled samples. *Advances in Neural Information Processing Systems*, 34:15895–15907, 2021.

[123] Li Zhang, Tao Xiang, and Shaogang Gong. Learning a deep embedding model for zero-shot learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2021–2030, 2017.

[124] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

[125] Eyke Hüllermeier and Willem Waegeman. Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Machine Learning*, 110(3):457–506, 2021.

[126] Armen Der Kiureghian and Ove Ditlevsen. Aleatory or epistemic? does it matter? *Structural safety*, 31(2):105–112, 2009.

[127] Neil Houlsby, Ferenc Huszár, Zoubin Ghahramani, and Máté Lengyel. Bayesian active learning for classification and preference learning. *arXiv preprint arXiv:1112.5745*, 2011.

[128] Elmar Haussmann, Michele Fenzi, Kashyap Chitta, Jan Ivanecky, Hanson Xu, Donna Roy, Akshita Mittel, Nicolas Koumchatzky, Clement Farabet, and Jose M Alvarez. Scalable active learning for object detection. In *2020 IEEE intelligent vehicles symposium (iv)*, pages 1430–1435. IEEE, 2020.

[129] Burr Settles. Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2009.

[130] Vu-Linh Nguyen, Mohammad Hossein Shaker, and Eyke Hüllermeier. How to measure uncertainty in uncertainty sampling for active learning. *Machine Learning*, 111(1):89–122, 2022.

[131] Heinrich Jiang and Maya Gupta. Minimum-margin active learning. *arXiv preprint arXiv:1906.00025*, 2019.

[132] Jingyu Shao, Qing Wang, and Fangbing Liu. Learning to sample: an active learning framework. In *2019 IEEE International Conference on Data Mining (ICDM)*, pages 538–547. IEEE, 2019.

[133] Boshuang Huang, Sudeep Salgia, and Qing Zhao. Disagreement-based active learning in online settings. *arXiv preprint arXiv:1904.09056*, 2019.

[134] Andrew McCallum, Kamal Nigam, et al. Employing em and pool-based active learning for text classification. In *ICML*, volume 98, pages 350–358. Citeseer, 1998.

[135] Prem Melville, Stewart M Yang, Maytal Saar-Tsechansky, and Raymond Mooney. Active learning for probability estimation using jensen-shannon divergence. In *Machine Learning: ECML 2005: 16th European Conference on Machine Learning, Porto, Portugal, October 3-7, 2005. Proceedings 16*, pages 268–279. Springer, 2005.

[136] Jingbo Zhu, Huizhen Wang, Tianshun Yao, and Benjamin K Tsou. Active learning with sampling by uncertainty and density for word sense disambiguation and text classification. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 1137–1144, 2008.

[137] Ana Maria Vallina, Hanns de la Fuente-Mella, Jose Barrera, and Hugo Mansilla. Active learning methods to enhance higher education in business. In *13th International Conference on Applied Human Factors and Ergonomics (AHFE 2022)*, 01 2022.

[138] Zhi-Hua Zhou. A brief introduction to weakly supervised learning. *National science review*, 5(1):44–53, 2018.

[139] Massih-Reza Amini, Vasilii Feofanov, Loic Pauletto, Emilie Devijver, and Yury Maximov. Self-training: A survey. *arXiv preprint arXiv:2202.12040*, 2022.

[140] Giannis Karamanolakis, Subhabrata Mukherjee, Guoqing Zheng, and Ahmed Hassan Awadallah. Self-training with weak supervision. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tur, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou, editors, *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 845–863, Online, June 2021. Association for Computational Linguistics.

[141] Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

[142] Changyu Deng, Xunbi Ji, Colton Rainey, Jianyu Zhang, and Wei Lu. Integrating machine learning with human knowledge. *Iscience*, 23(11):101656, 2020.

[143] Marta Garnelo and Murray Shanahan. Reconciling deep learning with symbolic artificial intelligence: representing objects and relations. *Current Opinion in Behavioral Sciences*, 29:17–23, 2019.

[144] Erik Derner, Jiří Kubalík, and Robert Babuška. Selecting informative data samples for model learning through symbolic regression. *IEEE Access*, 9:14148–14158, 2021.

[145] Casper Wilstrup and Jaan Kasak. Symbolic regression outperforms other

models for small data sets. *arXiv preprint arXiv:2103.15147*, 2021.

[146] Feihong Wu, Jun Zhang, and Vasant Honavar. Learning classifiers using hierarchically structured class taxonomies. In *Abstraction, Reformulation and Approximation: 6th International Symposium, SARA 2005, Airth Castle, Scotland, UK, July 26-29, 2005. Proceedings 6*, pages 313–320. Springer, 2005.

[147] Guangzhou An, Masahiro Akiba, Kazuko Omodaka, Toru Nakazawa, and Hideo Yokota. Hierarchical deep learning models using transfer learning for disease detection and classification based on small number of medical images. *Scientific reports*, 11(1):4250, 2021.

[148] Abdolmaged Alkhulaifi, Fahad Alsahli, and Irfan Ahmad. Knowledge distillation in deep learning and its applications. *PeerJ Computer Science*, 7:e474, 2021.

[149] Jaemin Yoo, Minyong Cho, Taebum Kim, and U Kang. Knowledge extraction with no observable data. *Advances in Neural Information Processing Systems*, 32, 2019.

[150] Samuel Stanton, Pavel Izmailov, Polina Kirichenko, Alexander A Alemi, and Andrew G Wilson. Does knowledge distillation really work? *Advances in Neural Information Processing Systems*, 34:6906–6919, 2021.

[151] Anne-Katrin Mahlein, Erich-Christian Oerke, Ulrike Steiner, and Heinz-Wilhelm Dehne. Recent advances in sensing plant diseases for precision crop protection. *European Journal of Plant Pathology*, 133(1):197–209, 2012.

[152] Yeen Ting Hwang, Champa Wijekoon, Melanie Kalischuk, Dan Johnson, Ron Howard, Dirk Prüfer, and Lawrence Kawchuk. Evolution and management of the irish potato famine pathogen phytophthora infestans in canada and the united states. *American Journal of Potato Research*, 91, 2014.

[153] Julio M. Duarte-Carvajalino, Diego F. Alzate, Andrés A. Ramirez, Juan D. Santa-Sepulveda, Alexandra E. Fajardo-Rojas, and Mauricio Soto-Suárez. Evaluating late blight severity in potato crops using unmanned aerial vehicles and machine learning algorithms. *Remote Sensing*, 10, 2018.

[154] Laura Beggel, Michael Pfeiffer, and Bernd Bischl. Robust anomaly detection in images using adversarial autoencoders. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2019, Würzburg, Germany, September 16–20, 2019, Proceedings, Part I*, pages 206–222. Springer, 2020.

[155] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. Class-balanced loss based on effective number of samples. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9268–9277, 2019.

[156] Mahbub Hussain, Jordan J Bird, and Diego R Faria. A study on CNN transfer learning for image classification. In *UK Workshop on Computational Intelligence*, pages 191–202. Springer, 2018.

[157] David Hughes, Marcel Salathé, et al. An open access repository of images on plant health to enable the development of mobile disease diagnostics. *arXiv preprint arXiv:1511.08060*, 2015.

[158] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? In *Advances in Neural Information Processing Systems*, pages 2483–2493, 2018.

[159] Nils Bjorck, Carla P Gomes, Bart Selman, and Kilian Q Weinberger. Understanding batch normalization. In *Advances in Neural Information Processing Systems*, pages 7694–7705, 2018.

[160] Dmytro Mishkin and Jiri Matas. All you need is a good init. *arXiv preprint arXiv:1511.06422*, 2015.

[161] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[162] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

[163] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 3–19, 2018.

[164] Saurabh Singh and Shankar Krishnan. Filter response normalization layer: Eliminating batch dependence in the training of deep neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11237–11246, 2020.

[165] Yan Wang, Xiaofu Wu, Yuanyuan Chang, Suofei Zhang, Quan Zhou, and Jun Yan. Batch normalization: Is learning an adaptive gain and bias necessary? In *Proceedings of the 2018 10th International Conference on Machine Learning and Computing*, pages 36–40, 2018.

[166] Jonathan Frankle, David J Schwab, and Ari S Morcos. Training batchnorm and only batchnorm: On the expressive power of random features in CNNs. *arXiv preprint arXiv:2003.00152*, 2020.

[167] Qiuyu Zhu, Zikuang He, Tao Zhang, and Wennan Cui. Improving classification performance of softmax loss function based on scalable batch-normalization. *Applied Sciences*, 10(8):2950, 2020.

[168] Yen-Chang Hsu, Yilin Shen, Hongxia Jin, and Zsolt Kira. Generalized odin: Detecting out-of-distribution image without learning from out-of-distribution data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10951–10960, 2020.

[169] Shiyu Liang, Yixuan Li, and Rayadurgam Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. In *Proceedings of 6th International Conference on Learning Representations, ICLR 2018*, 2018.

[170] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[171] Sharada P Mohanty, David P Hughes, and Marcel Salathé. Using deep learning for image-based plant disease detection. *Frontiers in plant science*, 7:1419, 2016.

[172] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proceedings of International Conference on Learning Representations, ICLR 2015*, 2015.

[173] B. Zhou, A. Khosla, Lapedriza. A., A. Oliva, and A. Torralba. Learning Deep Features for Discriminative Localization. *CVPR*, 2016.

[174] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[175] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

[176] Anders Krogh and John A Hertz. A simple weight decay can improve generalization. In *Advances in neural information processing systems*, pages 950–957, 1992.

[177] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *Proceedings of International Conference on Learning Representations, ICLR 2014*, 2014.

[178] Yoshua Bengio et al. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

[179] Rafael Müller, Simon Kornblith, and Geoffrey E Hinton. When does label smoothing help? In *Advances in Neural Information Processing Systems*, pages 4696–4705, 2019.

[180] Ivan Chelombiev, Conor Houghton, and Cian O'Donnell. Adaptive estimators show information compression in deep neural networks. *arXiv preprint arXiv:1902.09037*, 2019.

[181] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1321–1330. JMLR. org, 2017.

[182] Glenn W Brier. Verification of forecasts expressed in terms of probability. *Monthly weather review*, 78(1):1–3, 1950.

[183] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. In *Advances in neural information processing systems*, pages 971–980, 2017.

[184] David Gutman, Noel CF Codella, Emre Celebi, Brian Helba, Michael Marchetti, Nabin Mishra, and Allan Halpern. Skin lesion analysis toward melanoma detection: A challenge at the international symposium on biomedical imaging (isbi) 2016, hosted by the international skin imaging collaboration (isic). *arXiv preprint arXiv:1605.01397*, 2016.

[185] Lei Zhang, Fan Yang, Yimin Daniel Zhang, and Ying Julie Zhu. Road crack detection using deep convolutional neural network. In *2016 IEEE international conference on image processing (ICIP)*, pages 3708–3712. IEEE, 2016.

[186] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059, 2016.

[187] Ashish Jaiswal, Ashwin Ramesh Babu, Mohammad Zaki Zadeh, Debapriya Banerjee, and Fillia Makedon. A survey on contrastive self-supervised learning. *Technologies*, 9(1):2, 2020.

[188] Robert Epstein. The empty brain: Your brain does not process information and it is not a computer. *Aeon Essays*, 2016.

[189] Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. Pre-trained models for natural language processing: A survey. *Science China Technological Sciences*, 63(10):1872–1897, 2020.

[190] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[191] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of naacL-HLT*, volume 1, page 2, 2019.

[192] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[193] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 297–304. JMLR Workshop and Conference Proceedings, 2010.

[194] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26, 2013.

[195] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[196] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in Neural Information Processing Systems*, 33:21271–21284, 2020.

[197] Stavros P Adam, Stamatios-Aggelos N Alexandropoulos, Panos M Pardalos, and Michael N Vrahatis. No free lunch theorem: A review. *Approximation and Optimization: Algorithms, Complexity and Applications*, pages 57–82, 2019.

[198] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*,

pages 9729–9738, 2020.

[199] Hiroki Nakamura, Masashi Okada, and Tadahiro Taniguchi. Self-supervised representation learning as multimodal variational inference. *arXiv preprint arXiv:2203.11437*, 2022.

[200] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *Advances in Neural Information Processing Systems*, 33:9912–9924, 2020.

[201] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *International Conference on Machine Learning*, pages 12310–12320. PMLR, 2021.

[202] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9650–9660, 2021.

[203] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020.

[204] Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9640–9649, 2021.

[205] Jovana Mitrovic, Brian McWilliams, and Melanie Rey. Less can be more in contrastive learning. In Jessica Zosa Forde, Francisco Ruiz, Melanie F. Pradier, and Aaron Schein, editors, *Proceedings on "I Can't Believe It's Not Better!" at NeurIPS Workshops*, volume 137 of *Proceedings of Machine Learning Research*, pages 70–75. PMLR, 12 Dec 2020.

[206] Daesoo Lee and Erlend Aune. Vibcreg: Variance-invariance-better-covariance regularization for self-supervised learning on time series. *arXiv preprint arXiv:2109.00783*, 2021.

[207] Souradip Chakraborty, Aritra Roy Gosthipaty, and Sayak Paul. G-simclr: Self-supervised contrastive learning with guided projection via pseudo labelling. In *2020 International Conference on Data Mining Workshops (ICDMW)*, pages 912–916. IEEE, 2020.

[208] Ali Borji, Ming-Ming Cheng, Qibin Hou, Huaizu Jiang, and Jia Li. Salient object detection: A survey. *Computational visual media*, 5(2):117–150, 2019.

[209] Marius Leordeanu. *Unsupervised Learning in Space and Time*, volume 1. Springer, 2020.

[210] Yuhua Chen, Cordelia Schmid, and Cristian Sminchisescu. Self-supervised learning with geometric constraints in monocular video: Connecting flow, depth, and camera. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7063–7072, 2019.

[211] Ozan Ciga and Anne L Martel. Learning to segment images with classification labels. *Medical Image Analysis*, 68:101912, 2021.

[212] Saleh Albelwi. Survey on self-supervised learning: Auxiliary pretext tasks and contrastive learning methods in imaging. *Entropy*, 24(4):551, 2022.

[213] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2536–2544, 2016.

[214] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *European conference on computer vision*, pages 649–666. Springer, 2016.

[215] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European conference on computer vision*, pages 69–84. Springer, 2016.

[216] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a

generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4681–4690, 2017.

[217] Dejing Xu, Jun Xiao, Zhou Zhao, Jian Shao, Di Xie, and Yueting Zhuang. Self-supervised spatiotemporal learning via video clip order prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10334–10343, 2019.

[218] Triantafyllos Afouras, Yuki M Asano, Francois Fagan, Andrea Vedaldi, and Florian Metze. Self-supervised object detection from audio-visual correspondence. *arXiv preprint arXiv:2104.06401*, 2021.

[219] Abhinav Dhere and Jayanthi Sivaswamy. Self-supervised learning for segmentation. *arXiv preprint arXiv:2101.05456*, 2021.

[220] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey E Hinton. Big self-supervised models are strong semi-supervised learners. *Advances in neural information processing systems*, 33:22243–22255, 2020.

[221] Alexander Kolesnikov, Xiaohua Zhai, and Lucas Beyer. Revisiting self-supervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1920–1929, 2019.

[222] Randall Balestriero, Leon Bottou, and Yann LeCun. The effects of regularization and data augmentation are class dependent. *Advances in Neural Information Processing Systems*, 35:37878–37891, 2022.

[223] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. " grabcut" interactive foreground extraction using iterated graph cuts. *ACM transactions on graphics (TOG)*, 23(3):309–314, 2004.

[224] Ming-Ming Cheng, Niloy J Mitra, Xiaolei Huang, Philip HS Torr, and Shi-Min Hu. Global contrast based salient region detection. *IEEE transactions on pattern analysis and machine intelligence*, 37(3):569–582, 2014.

[225] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[226] Xu Ji, Joao F Henriques, and Andrea Vedaldi. Invariant information clustering for unsupervised image classification and segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9865–9874, 2019.

[227] Asako Kanezaki. Unsupervised image segmentation by backpropagation. In *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 1543–1547. IEEE, 2018.

[228] Wonjik Kim, Asako Kanezaki, and Masayuki Tanaka. Unsupervised learning of image segmentation based on differentiable feature clustering. *IEEE Transactions on Image Processing*, 29:8055–8068, 2020.

[229] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. Let there be color! joint end-to-end learning of global and local image priors for automatic image colorization with simultaneous classification. *ACM Transactions on Graphics (ToG)*, 35(4):1–11, 2016.

[230] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Learning representations for automatic colorization. In *European conference on computer vision*, pages 577–593. Springer, 2016.

[231] Hou Jiang, Ling Yao, Ning Lu, Jun Qin, Tang Liu, Yujun Liu, and Chenghu Zhou. Multi-resolution dataset for photovoltaic panel segmentation from satellite and aerial imagery. *Earth System Science Data*, 13(11):5389–5401, 2021.

[232] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, 2009.

[233] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 215–223.

JMLR Workshop and Conference Proceedings, 2011.

[234] Mark Hamilton, Zhoutong Zhang, Bharath Hariharan, Noah Snavely, and William T Freeman. Unsupervised semantic segmentation by distilling feature correspondences. *arXiv preprint arXiv:2203.08414*, 2022.

[235] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[236] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019.

[237] Elijah Cole, Xuan Yang, Kimberly Wilber, Oisin Mac Aodha, and Serge Belongie. When does contrastive visual representation learning work? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14755–14764, 2022.

English Summary

Learning from small data sets in machine learning is a crucial challenge, especially when dealing with data imbalances and anomaly detection. This thesis delves into the challenges and methodologies of learning from small datasets in machine learning, with a particular focus on addressing data imbalances and anomaly detection. It thoroughly explores various strategies for effective small dataset learning in ML, examining both existing approaches and introducing novel techniques. The research pivots around two key questions: firstly, it investigates current methods employed for learning from small datasets in machine learning, and secondly, it assesses the efficacy of batch normalization in enhancing model performance and utilizing salient image segmentation as an augmentation policy in self-supervised learning.

The thesis comprehensively reviews techniques for managing small datasets, including data selection and preprocessing, ensemble methods, transfer learning, regularization techniques, and synthetic data generation. A critical examination of batch normalization reveals its significant role in improving training time and testing errors for minority classes in highly imbalanced datasets. The study also demonstrates that utilizing salient image segmentation as an augmentation policy in self-supervised learning substantially improves representation learning. This improvement is particularly evident in the context of downstream tasks such as image segmentation, highlighting the effectiveness of this technique in enhancing model performance.

In summary, this study contributes to the field of machine learning by exploring

strategies for learning from small datasets. It offers a detailed analysis of batch normalization, highlighting its potential in improving performance for minority classes in imbalanced datasets. Additionally, the study introduces salient image segmentation as an augmentation policy in self-supervised learning, showing its effectiveness in tasks like image segmentation. These findings provide a solid foundation for further research in small sample learning and present practical insights for machine learning practitioners working with limited data.

Nederlandse Samenvatting

Kleine datasets vormen een uitdaging voor veel machine learning technieken, in het bijzonder bij ongebalanceerde datasets en anomaliteiten detectie. Dit proefschrift onderzoekt verschillende benaderingen voor het effectief leren van kleine datasets, het aanpakken van uitdagingen. Daarnaast biedt het een uitgebreide analyse van bestaande technieken in de literatuur. Het onderzoek richt zich specifiek op twee hoofdvragen: 1) wat zijn de huidige methoden voor het leren van taken gebaseerd op kleine datasets in machine learning, en 2) wat is de impact van batch normalisatie en opvallende beeldsegmentatie als augmentatie techniek bij self-supervised learning op de prestaties van het model.

Het proefschrift presenteert een gedetailleerd overzicht van technieken voor het omgaan met kleine datasets, zoals data selectie, data voorverwerking, ensemble-methoden, transfer learning, regularisatie technieken en synthetische data generatie. Vervolgens wordt de impact van batch normalisatie onderzocht, waarbij wordt aangetoond dat deze de trainingstijd en de kwaliteit van de voorspellingen voor minder voorkomende klassen in sterk ongebalanceerde datasets aanzienlijk verbetert. Het onderzoek gaat verder in op de rol van self-supervised learning als augmentatie techniek, waarbij wordt aangetoond dat het gebruik van opvallende beeldsegmentatie de geleerde representaties verbetert, vooral in downstream taken zoals beeldsegmentatie.

Samenvattend draagt deze studie bij aan het vakgebied van machine learning door strategieën te verkennen voor het leren van kleine datasets. Het biedt een gedetailleerde analyse van batch normalisatie, waarbij het potentieel wordt be-

nadrukt om de prestaties voor minderheidsklassen in ongebalanceerde datasets te verbeteren. Daarnaast introduceert de studie opvallende beeldsegmentatie als augmentatiebeleid in zelfgestuurd leren, waarbij de effectiviteit ervan in taken zoals beeldsegmentatie wordt aangetoond. Deze bevindingen bieden een solide basis voor verder onderzoek in het leren van taken gebaseerd op kleine datasets en presenteren praktische inzichten voor machine learning professionals die werken met beperkte data.

Acknowledgements

Embarking on this PhD journey has been an extraordinary chapter in my life, marked by immense challenges and profound growth, all made possible by the support and belief of remarkable individuals.

I am profoundly grateful to Prof. Thomas Bäck, who saw potential in me from the very beginning. Despite the complexities of my personal and professional journey, including a daunting relocation to a new country, Prof. Bäck's unwavering faith in me was a guiding light. His mentorship, particularly as I navigated the unfamiliar terrains of academia as one of his most senior students, provided me with the courage and space to flourish. His encouragement during times of doubt was invaluable, and for that, I am eternally grateful.

Equally, my heartfelt thanks go to Dr. Ofer Shir, my co-advisor, and friend. His dedication transcended physical distances, as he meticulously guided me from thousands of miles away with unwavering patience and wisdom. Dr. Shir's thorough reviews and constructive feedback transformed my nascent ideas into academic pursuits. His calm and positive approach was a constant source of inspiration, shaping the very essence of my research.

To my family – my wife and two sons – who not only journeyed with me but also endured significant sacrifices for this dream. Our move from Turkey to pursue this PhD was a collective leap of faith, profoundly altering our family life. My most heartfelt appreciation goes to my wife, whose resilience and unwavering support have been the cornerstone of my journey. Her immense sacrifice, especially during

the countless holidays and weekends when my research kept me away, is a testament to her extraordinary strength and love. I am deeply grateful for her and my sons' understanding and patience as they coped with my frequent absences, a challenging but crucial part of our shared journey.

My siblings, who have always been my cheerleaders, deserve my heartfelt thanks. Their belief in my ambitions, their endless love and support, and their willingness to listen, have been pillars of my emotional strength.

To my dear father, whose absence in the final stages of my PhD journey has been a profound sorrow in my heart. His passing, mere months before the culmination of my efforts, leaves a void that words cannot fill. For the last few years, distance and my work kept us apart, and I had always envisioned him proudly witnessing my achievement. Although fate had other plans, his memory and spirit have been a guiding light, even in his absence. I hope he is watching over me with pride, as I reach this milestone in my life. His love and presence, though now a cherished memory, continue to inspire and strengthen me.

Finally, I extend my gratitude to my employers and business partners throughout my PhD journey. Their understanding and encouragement in balancing my academic pursuits with professional commitments were instrumental in making this journey possible.

This PhD has been more than an academic endeavor; it has been a transformative experience, enriched and made possible by each of you. Thank you for being part of my journey.

About the Author

Veysel Kocaman was born in Adiyaman, Turkey on the 17th January 1981. After completing his undergraduate studies in Computer Engineering in 2003, Veysel's passion for technology and innovation led him to pursue further education in Operations Research & Industrial Engineering at Penn State University, USA, where he graduated with a Master of Science degree in 2009. His academic journey culminated with a Ph.D. in Computer Science from Leiden Institute of Advanced Computer Science (LIACS), Leiden University, Netherlands, in 2024, under the supervision of Prof.Dr. T.H.W. Bäck, reflecting his deep commitment to advancing knowledge in his field.

During his Ph.D., Veysel focused on self-supervised learning strategies and specialized in developing advanced neural network architectures tailored for effective learning in environments characterized by limited sample sizes and significant dataset imbalances, a pursuit that culminated in his dissertation titled 'Learning From Small Samples'. His expertise is particularly pronounced in applying computer vision and NLP methods within the Healthcare and Agriculture domains.