Dublin City University
School of Electronic Engineering

Visual Representation Learning with Deep Neural Networks Under Label and Budget Constraints

Eric Arazo Sánchez, B.E., M.E.

A Dissertation submitted in fulfilment of the requirements for the award of Doctor of Philosophy (Ph.D.)

Supervised by Dr Kevin McGuinness
Co-supervised by Professor Noel E. O'Connor

December 2021

Declaration

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Doctor of Philosophy is entirely my own work, and that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge breach any law of copyright, and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Signed :  Student number: 16104871

Eric Arazo Sánchez

Date : 17^{th} of December 2021

Acknowledgements

First and foremost, I would like to express my gratitude to my supervisors, Prof. Kevin McGuinness and Prof. Noel O'Connor, for the opportunity to start this journey and the support to finish it. Thank you for your great mentorship; your example and advice will follow me for years to come. Thank you, Kevin, for the patience and wisdom with which you guided me through the darkest corners of machine learning. Your advice often reaches further than the Ph.D and seems to be the one I need at every moment. Thank you, Noel, for your excellent inputs and continuous support.

The completion of this dissertation would not have been possible without the help of Dr. Diego Ortego and Paul Albert. Thank you, Diego, for your invaluable support and advice during this journey. These have been pivotal in my development as a researcher. I am particularly grateful for our fruitful discussions, which have lead to many interesting observations and late Friday afternoon conversations. Thanks, Paul, for the close collaboration we have had during these years. I hope I can also add some small sand grains to your Ph.D.

As a crucial contributor to my researcher life, I would like to thank Prof. Xavi Giro-i-Nieto for sharing his passion for deep learning and introducing me to the fascinating research that the Insight lab does under Noel and Kevin's supervision. Thanks also to Eva for guiding me through my landing on Insight, that together with Dian and Joseph, made the beginning of my Ph.D. less scary. I am also grateful to the colleagues and friends I have met in Insight, especially Camille, Enric, Lluis, and Jaime.

To my parents Javier and Marisa, my sibling Maria-Yael, and my grandmother Enriqueta: you have always encouraged and pushed me to overcome all the challenges that come with life. Thank you for being patient, and thank you for understanding my "airplane mode" before deadlines. I hope that we spend some time together soon and we can reconnect with each other.

List of publications

Main publications:

- **Eric Arazo***, Diego Ortego*, Paul Albert, Noel E. O'Connor, Kevin McGuinness. "Unsupervised Label Noise Modeling and Loss Correction". In *International Conference on Machine Learning (ICML)*. June 2019.
- **Eric Arazo**, Noel E. O'Connor, Kevin McGuinness. "Improving Unsupervised Learning with ExemplarCNNs". In *Irish Machine Vision and Image Processing Conference (IMVIP)*. August 2019.
- **Eric Arazo**, Diego Ortego, Paul Albert, Noel E. O'Connor, Kevin McGuinness. "Pseudo-Labeling and Confirmation Bias in Deep Semi-Supervised Learning" *International Joint Conference on Neural Networks (IJCNN)*. June 2020.
- Diego Ortego, **Eric Arazo**, Paul Albert, Noel E. O'Connor, Kevin McGuinness. "Towards Robust Learning with Different Label Noise Distributions". In *International Conference on Pattern Recognition (ICPR)*. January 2021.
- Diego Ortego, **Eric Arazo**, Paul Albert, Noel E. O'Connor, Kevin McGuinness. "Multi-Objective Interpolation Training for Robustness to Label Noise". In *Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2021.
- Paul Albert, Diego Ortego, **Eric Arazo**, Noel E. O'Connor, Kevin McGuinness. "ReLaB: Reliable Label Bootstrapping for Semi-Supervised Learning". In *International Joint Conference on Neural Networks (IJCNN)*. July 2021.
- **Eric Arazo**, Diego Ortego, Paul Albert, Noel E. O'Connor, Kevin McGuinness. "How Important is Importance Sampling for Deep Budgeted Training?" *British Machine Vision Conference (BMVC)*. November 2021.
- Paul Albert, Diego Ortego, **Eric Arazo**, Noel E. O'Connor, Kevin McGuinness. "Addressing out-of-distribution label noise in webly-labelled data". In *Winter Conference on Applications of Computer Vision (WACV)*. January 2022.

Other publications:

- Diego Ortego, Kevin McGuinness, Juan C SanMiguel, **Eric Arazo**, José M Martínez, Noel E. O'Connor. "On guiding video object segmentation". In *International Conference on Content-Based Multimedia Indexing (CBMI)*. September 2019.

- Alan F. Smeathon, Yvette Graham, Kevin McGuinness, Noel E. O'Connor, Seán Quinn, **Eric Arazo**. "Exploring the Impact of Training Data Bias on Automatic Generation of Video Captions". In *International Conference on Multimedia Modeling (MMM)*. December 2018.

- Haithem Afli, Feiyan Hu, Jinhua Du, Daniel Cosgrove, Kevin McGuinness, Noel E. O'Connor, **Eric Arazo**, Jiang Zhou, Alan Smeaton. "Dublin City University Participation in the VTT Track at TRECVid 2017". In *TRECVid Workshop*. November 2017.

(*) Equal contribution.

Contents

List of Tables

List of Figures

Acronyms and Abbreviations

This section provides a glossary of the main acronyms and abbreviations used in the thesis:

NN	Neural Network
CNN	Convolutional Neural Network
DNN	Deep Neural Network
GD	Gradient Descent
SGD	Stochastic Gradient Descent
CE	Cross-Entropy
OOD	Out-of-distribution
ID	In-distribution
GMM	Gaussian Mixture Model
BMM	Beta Mixture Model
AUC	Area Under the Curve
EM	Expectation Maximization
RN	ResNet
WR	Wide-ResNet
PR	Pre-activation-ResNet
MSE	Mean Square Error
ReLU	Rectified Linear Unit
PCA	Principal Component Analysis
HSV	Hue Saturation Value

RGB Red Green Blue

SVM Support Vector Machine

HAC Hierarchical Agglomerative Clustering

k-NN k-Nearest Neighbours

NCE Noise-Contrastive Estimation

GPU Graphic Processing Unit

Mathematical notation

This section provides a glossary of the notation used through the thesis, which closely follows the notation used in [21]:

x	Scalars are denoted in italic lowercase Roman or Greek letters.
$\mathbf{x} = (x_1, ...x_n)^T$	Vectors are denoted in lower case bold Roman letters and are assumed to be column vectors. Note that the elements inside a vector are contained between brackets and that $(x_1, x_2, ...x_n)$ denotes the corresponding row vector $\mathbf{x}^T$ with n scalars.
$\mathbf{x}^T$	The superscript T indicates the transpose of a vector, matrix, or tensor.
X	Matrices are denoted with uppercase italic Roman letters and the element X_{ij} corresponds to the row i and column j. Tensors are also denoted with uppercase italic Roman letters.
$X = \{x_1, ..., x_n\}$	Sets are denoted with uppercase italic Roman letters and the enclosed elements are contained between curly braces.
$f(x)$	Function are denoted with lowercase italic Roman letters, e.g. $f(x)$ is a function of x.

Abstract

"Visual Representation Learning with Deep Neural Networks Under Label and Budget Constraints"
Eric Arazo Sánchez

This thesis presents the work done in the area of semi-supervised learning, label noise, and budgeted training for deep learning approaches to computer vision. The improvements seen in computer vision since the successful introduction of deep learning rely on the availability of large amounts of labeled data and long lasting training processes. First, this research studies the three main alternatives to fully supervised deep learning categorized in three different levels of supervision: unsupervised learning (no label involved), semi-supervised learning (a small set of labeled data is available), and label noise (all the samples are labeled but some of them are incorrect). These alternatives aim at reducing the cost of building fully annotated and finely curated datasets, which in most cases is time consuming and requires expert annotators. State-of-the-art performance has been achieved in several semi-supervised, unsupervised, and label noise benchmarks including CIFAR10, CIFAR100, and STL-10. Additionally, the solutions proposed for learning in the presence of label noise have been validated in realistic benchmarks built with datasets annotated from web information: WebVision and Clothing1M. Second, this research explores alternatives to reduce the computational cost of the training of deep learning systems that currently require hours or days to reach state-of-the-art performance. Particularly, this research studied budgeted training, i.e. when the training process is limited to a fixed number of iterations. Experiments in this setup showed that for better model convergence, variety in the data is preferable than the importance of the samples used during training. As a result of this research, three main author publications have been generated, one more has been recently submitted to review for a conference, and several other secondary author publications have been produced in close collaboration with other researchers in the centre.

Chapter 1

Introduction

Computer vision is the research field that enables the processing of visual data to extract useful information and gain high-level understanding of the world. The practical aim is to provide visual sensing autonomy to computer systems to interact with the world and perform tasks that include face recognition, medical image analysis, or autonomous driving. To this end, computer vision leverages techniques and practices from the field of machine learning and, in a more scientific aspect, overlaps with other research fields such as statistics, optimization, and signal processing to understand and develop the theory behind digital systems that explore visual information.

Research in this area has experienced a noticeable transformation in recent years, particularly regarding algorithm performance and its underlying principles. These improvements have been mainly fostered by large amounts of data becoming more easily available, technological advances allowing for the development of more complex systems (regarding memory and computational speed), and the successful application of neural networks that enable the extraction of representative features from real-world data.

The aim of this thesis is to explore neural networks as a tool for learning visual representations[1]and focuses on the alternatives to avoid the two main challenges

[1]Representations of visual concepts in high-dimensional spaces are often referred to as visual representation. These are projections of input samples (e.g. images) in a space defined by the parameters of the given model.

from current neural network-based visual systems: the need for large amounts of carefully annotated data and the computational cost that comes with it. This chapter provides an overview and the motivation of the research presented in this thesis, states the hypotheses and the main research questions, and lays out the structure of the document.

1.1 Learning representations with neural networks

Research in representation learning dates back to the 1950s, when neural networks were invented. As biologically inspired parametric functions, neural networks brought to machine learning the idea of learning representations directly from the data. From their invention until the beginning of the last decade, neural network research had several waves of interest, becoming more popular after every breakthrough. During this period, the field overcame each of the challenges that made neural networks unpractical to apply in realistic scenarios: lack of an efficient learning algorithm, low memory and computational power, or small amounts of data available. Consequently, at the beginning of the 2000s, neural networks were being used in speech recognition systems and later in 2012 were successfully applied for the first time to large-scale computer vision applications. Initially they were applied to image classification but soon their popularity spread to more complex applications such as semantic segmentation, object localization, or face recognition.

Before neural networks, models used in machine learning applications consisted of two main blocks: a feature extractor and a classifier. The feature extractor was a module that searched for certain manually-predefined features in the data. This accounted for the main part of the model: better features would describe the information in the data more accurately. The classifier on the other hand was a relatively small block that mapped the features to the target task. This was not handcrafted, but trained on a subset of the data. Neural networks shifted this paradigm and allowed for completely data-driven models where all the parameters were randomly initialized and optimized by iterating through the training data.

1.1.1 High level overview of deep learning

Neural networks consist of a composition of layers, each mapping its input to an increasing level of abstraction through linear and non-linear operations (involving matrix multiplications and non-linear functions): from the lowest level in the input space (e.g. images) to the highest in the output space (e.g. classes). All these layers constitute a single block, trainable end-to-end with very little human intervention. This concept of stacking layers upon layers and training them from the data gave rise to the name of *deep learning*, and in most cases, outperformed all hand-crafted alternatives [128, 133, 52] (improving performance as data increased).

The underlying principle behind neural networks is based on empirical risk minimization[2]: the parameters are updated based on the performance of the model in a subset of the data designated for training. More concretely, the model is presented with batches of samples and its parameters are updated in a direction selected to reduce the average prediction error in those samples. This process is done several times over the dataset until convergence to a reasonable error value (often visiting all the samples several times). Chapter 2 presents the technical details and basic concepts behind the training of neural networks. Note that the parameters that are not learned through empirical risk minimization and are manually tuned are called hyperparameters, e.g. number of training iterations or number of layers in a neural network.

1.1.2 Neural networks for computer vision

One of the most relevant drives of the advances in the application of neural networks to computer vision tasks was the ImageNet competition [42]. It provided a setup for researchers to benchmark approaches for image classification (among other tasks)

[2] A "risk" value is often defined as a theoretical bound on the algorithm performance for evaluation purposes. In the training set, this is referred to as the "empirical risk," which is often called the "cost" or "loss" value, and serves as a metric to quantify the error of the model. However, since the true data distribution is unknown, this value – the "true risk" in this case – is also unknown. The empirical risk minimization principle assumes that minimizing the "empirical risk," will also result in a low value of the "true risk."

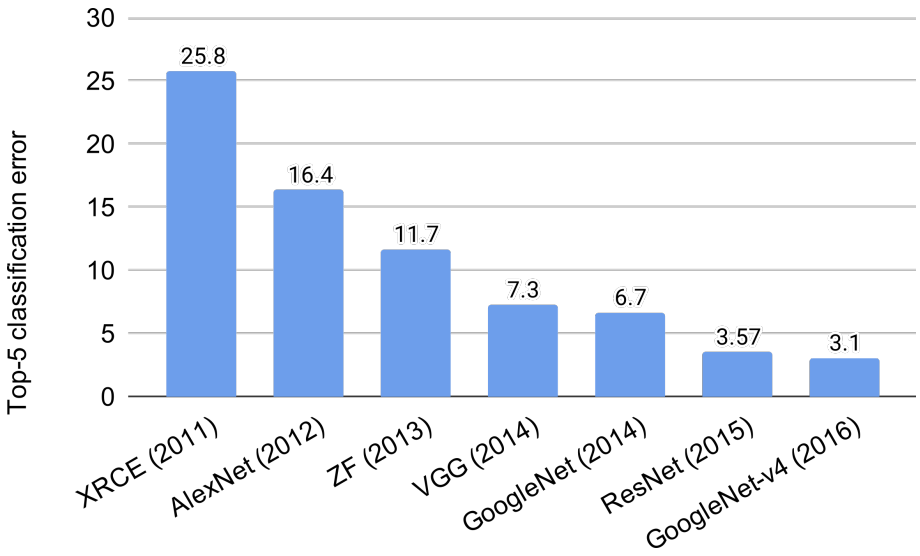


Figure 1.1: Evolution of the top-5 error of the classification task in ImageNet across time [131, 72]. Note that the top-5 error considers the prediction as correct if the true label is among the five categories predicted with highest probability.

and the largest set of finely curated data. The full dataset is approximately 14M images and the subset commonly used is a set of 1.2 million of images distributed across 1 000 categories that are organized accordingly to WordNet (a lexical database for English) [121]. This dataset remains of great importance to the computer vision community; it is a common practice to pretrain neural networks on ImageNet before fine-tuning them in the target task to obtain a better initialization.

The history of ImageNet and neural networks cross paths for the first time in the competition of 2012. On this occasion Alex Krizhevsky and his team [1] won the classification task with a neural network-based approach; this was the first time a graphic processing unit (GPU) was used to train a neural network in a large-scale visual dataset. This was not only a milestone for the advance on the technical aspect of training neural networks, but also a revolutionizing result for the competition: the proposed model reduced the error by 10%, while other teams were improving previous submissions to the challenge by a few percentage points (see Figure 1.1). Following that success, the implementation of Razavian *et al.* [158] outperformed most of the previous approaches on several other benchmarks by transferring the features learned on ImageNet and fine-tuning them on the target task. This result showed that neural

networks trained on ImageNet were very efficient feature extractors for non-related visual tasks (interestingly, these features are currently also being used in non-visual tasks [138]). Note that the neural networks commonly used in computer vision embed the basic principles behind convolution to leverage the spatial structure in images, which results in more robust representations and a reduction in the number of parameters in the model. These networks are called convolutional neural networks and are discussed in detail in Chapter 2.

The classification task might have been the one that drove the initial success of neural networks because of its simplicity: each image has to be associated with a single label. However, convolutional neural networks have shown extraordinary success in most of the computer vision tasks, including the most complex ones like panoptic segmentation (where each pixel of an image has to be classified to form different masks that classify and locate the objects in an image), pose estimation (were the body pose of different actors has to be predicted) or object detection (where bounding boxes allocate and classify the objects present on the image). Figure 1.2 shows some examples of applications of convolutional neural networks to computer vision tasks.

1.2 Motivation

The success of neural networks is tightly linked to the increasing availability of vast amounts of carefully annotated data and long-lasting training processes where samples are visited iteratively until convergence to a reasonable performance. These are also two significant challenges when training neural networks.

By visiting a large variety of examples during training, neural networks learn complex representations that are robust to visual variations of the input image and generalize to unseen samples. The data presented to the model, however, have to be exhaustively annotated: the presence of mislabeled samples or samples that do not belong to the expected distribution of classes can lead to degradation of the representations learned by the model. Unfortunately, the construction of large

Figure 1.2: Examples of computer vision tasks. Top from left to right: image classification [1], object detection [152], and pose estimation [170]. Bottom: semantic segmentation [144]

datasets is very costly and, as the amount of data increases, corruption becomes more likely to occur. In this scenario, models that learn robust representations when there are few labels available will help to reduce this dependence on the labelling process. Similarly, recent large-scale datasets leverage information from web searches and user tags to reduce the costs of the annotation process, resulting in extensive datasets with heavily corrupted annotations. In this case, models that learn robust representations under label noise conditions will be able to leverage these large amounts of easily accessible data.

This dependence of neural networks on the availability of finely labeled data presents a significant challenge when the end goal is practical real-world applications like autonomous driving, food quality estimation, or medical imaging, where the label availability is scarce (e.g. labels are difficult to obtain or annotations have to be done by experts). Consequently, robust methods to train under label constraints (few labels available or corrupted labels) are very important in these setups.

Aside from precise labels, learning complex representations requires typically in the order of a few hundred iterations through the full dataset. This translates into one or many GPUs (up to hundreds of them in some applications) computing matrix multiplications in parallel during hundreds or thousands of hours, and leads to the second drawback when training neural networks: long-lasting training processes. As the datasets and model size increase, so do the training times and power requirements. This is of utmost importance in situations where computational or time resources are limited, e.g. small companies, and where power consumption is a concern, e.g. mobile devices. Similarly, as noted by Anthony *et al.* [8], the strong environmental impact of training deep neural networks has the potential of making machine learning a significant contributor to climate change, e.g. the authors estimate, in appendix D, the energy and carbon footprint of training GPT-3 [25] (a model used in natural language processing applications) to be equivalent to over 700 000 km travelled by car. Additionally, aiming at more efficient training processes would benefit the applicability and spread of neural networks-based systems.

1.2.1 Representation learning under label and computational constraints

Solutions to reduce the dependence of neural networks on intensively annotated data aim at designing robust algorithms to train more efficiently in constrained scenarios where: no samples are labelled, only a small subset of samples are labelled, or the samples are incorrectly labelled with a certain probability. In the first scenario, neural networks train directly on images, dispensing with the labels completely. These solutions are discussed in Chapter 3 and are commonly referred to as "self-supervised" or "unsupervised" learning. The second scenario is "semi-supervised" learning, discussed in depth in Chapter 4, and only a small subset of the samples are labelled, reducing in this way the annotation requirements. Finally, in the third scenario the model trains directly with corrupted labels (often referred to as label noise training), which is discussed in Chapter 5.

A significant body of literature addresses solutions to reduce the computational costs of training neural networks through more efficient optimization algorithms, through methods to select the most relevant samples, or through strategies to organize the samples in a predefined more efficient curriculum. Despite ample work done in this direction, it is still unclear what are the benefits from these approaches when the training is constrained to a given budget of iterations. This setup, i.e. budgeted training, is explored in Chapter 6.

1.3 Hypothesis and research questions

Current research in computer vision is actively exploring the challenges introduced in this chapter in search of alternatives to reduce the need for carefully labelled data to train neural networks. However, the best way to capitalize on scarce or incorrect annotations remains under discussion. Previous to the work explored in this thesis, the existing methods failed to efficiently propagate the knowledge from a small subset of labelled data to a much larger unlabeled set and to separate clean

from noisy samples when they were blended together. Similarly, the literature that aims to accelerate the convergence of neural networks rarely studies the behavior of the proposed algorithms when the training is limited to a certain number of iterations. The following hypotheses have guided the development of this thesis towards exploring these two challenges in the training of neural networks: strong dependence on carefully labelled samples and large computational requirements.

H1: Given a large set of unlabeled images, clustering algorithms aid the self-supervised training of neural networks by enforcing the assumption that images belonging to the same class, or images with similar visual features, are closer than those that are from different classes when mapped to the representation space learned by the neural network.

- Initial approaches to self-supervised learning proposed to solve proxy tasks as an objective to train neural networks: solving jigsaw puzzles, predicting colorization, or prediction rotations are some well-known examples [132, 220, 53]. However, these approaches do not consider the clustering assumption (visually similar images should be mapped close in the embedding space learned by a neural network). Recent approaches to self-supervised learning use contrastive learning objectives that go in this direction [70, 29] but still ignore the relationships between classes and similarity between examples in the learned representation space.

H2: Neural networks can overcome the dependence on strong supervision from carefully annotated datasets by leveraging a trustworthy small subset of the data and extrapolating the features learned to the rest of the data.

- Existing research has shown that when there are few labelled samples in the dataset, the features learned by neural networks in the small labelled subset can be leveraged to exploit the unlabeled samples and improve feature quality [101]. Similarly, when the dataset is corrupted with label noise, the predictions

of the model can be used to correct the training and to detect a small subset of correctly annotated samples [150, 44]. In these cases, particularly when the reliable subset is several orders of magnitude smaller than the whole dataset, neural networks need strong regularization to reduce the risk of overfitting to the small trusted set.

H3: In scenarios where the training budget is restricted to a certain amount of computation, neural networks do not benefit from seeing the most important samples, but from seeing more variety in the data.

- While it has been proven that a certain sampling strategy is optimal for training neural networks, the cost of selecting these most important samples to use during training outweighs the reduction in training time [87]. In practical scenarios, when the training is restricted to a certain number of iterations, data augmentation is an effective solution to increases the variety of data shown to the network.

The following research questions have guided this thesis and shaped the exploration of the proposed hypotheses:

RQ1: How could the clustering assumption be enforced in the self-supervised training of neural networks? Could neural networks benefit from clustering the features learned during a self-supervised training?

RQ2: How can neural networks overcome the main limitation when propagating knowledge from a reliable subset of the data to the remaining larger subset? How can one determine which subset is reliable?

RQ3: Are there some samples more useful than others when training a neural network? Is it possible to leverage them to achieve better performance when the training is limited to a certain budget?

1.4 Thesis structure

The remainder of the thesis is structured as follows. In Chapter 2, we provide the technical background of the thesis and explore the related work. This chapter addresses the principles behind learning representations with neural networks and includes a review of the different datasets, experimental setup, and notation used in the rest of the thesis.

In Chapter 3, we present the work done in unsupervised learning through the proxy task of identifying examples from multiple views generated from each image and shows how this self-supervised approach can be improved when applying clustering techniques. This chapter also includes a discussion of the current methods for unsupervised learning through contrastive learning highlighting the main drawbacks and suggesting possible solutions based on the observations done in the experiments.

In Chapter 4 we explore how to propagate the knowledge learned in a small subset of labelled samples to a larger subset of unlabelled samples to further improve the representations learned. This chapter provides a thorough study of the hyperparameters involved and insights on the training of neural networks in a semi-supervised setup. Additionally, several model architectures are tested.

The focus of Chapter 5 is the label noise setup, where all the samples are labelled but some of them have incorrect labels. It proposes a method to identify incorrectly labelled samples and shows that the corresponding label predicted by the model can be used to guide the training and achieve a robust method to train neural networks under label noise. This chapter also introduces the different noise distributions to be expected in real-world datasets and provides a discussion on the robustness of different methods and noise detection metrics.

In Chapter 6 we assume proper supervision from the samples, i.e. all the samples are correctly labelled, and explore approaches to reduce the computational cost of training neural networks. This chapter presents a comparison of several methods from the literature under the same setup and experimentally shows the importance of the variability in the training data to reach a reasonable model performance.

Finally, Chapter 7 consists of an overview of the research presented in this thesis, a discussion of the hypotheses and research questions in light of the works presented, and suggestions of possible directions for further understanding the training of neural networks under label and computational constraints.

Chapter 2

Neural networks for computer vision

The first section of this chapter presents the background theory required for the following chapters, including the foundations of neural networks, the best practices adopted by the computer vision community when training them, and the assumptions adopted through the thesis. The literature review section then reviews the relevant approaches to the challenges introduced in Chapter 1. Finally, the methodology section introduces the datasets and setup from the experimental chapters of the thesis.

2.1 Introduction to neural networks for computer vision

During inference, neural networks map the input data to successive levels of abstraction until reaching the output space corresponding to the given task (this is often referred to as prediction stage or forward pass). In classification, for instance, this mapping generates a class prediction for each example. The comparison of these predictions to the ground truth annotations (the label associated with each example) results in a measure of the model prediction quality, which is often represented as

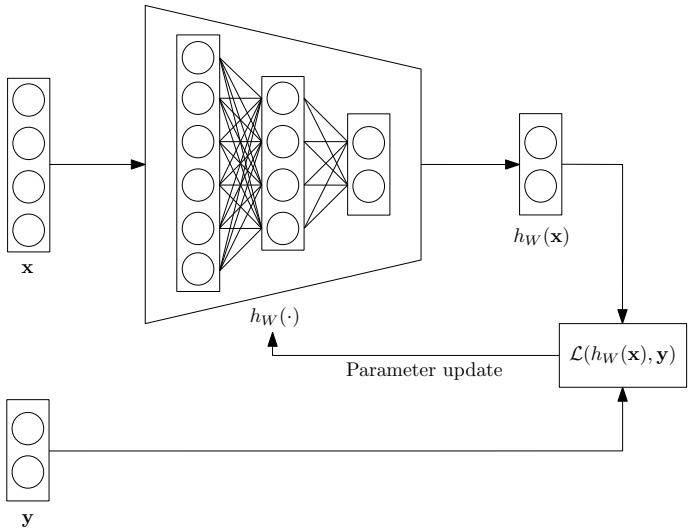


Figure 2.1: Schematic of backpropagation: $\mathbf{x}$ represents the input vector, $\mathbf{y}$ the ground-truth annotation (label), $h_w(\cdot)$ the neural network with w parameters, $h_w(\mathbf{x})$ the prediction of the network for the input $\mathbf{x}$, and $\mathcal{L}(\cdot)$ the loss function that computes the error value $\mathcal{L}(h_w(\mathbf{x}), \mathbf{y})$ used in the parameter update.

an error value. This error then is propagated backwards through the network – a process known as backpropagation – to guide the updates of the model parameters towards a configuration that minimizes the error value (Figure 2.1). This subsection describes in detail the different components of this process: the basic structure of neural networks, the optimization of their parameters, and the most widely used neural network architectures.

2.1.1 Neural networks and convolutional neural networks

Neural networks (NN) are compositions of several blocks, often referred to as layers. The most important of these layers performs two basic operations: a matrix multiplication followed by a non-linear activation function. On top of these two operations there are several other layers that contribute to the training and generalization of the model that for simplicity in the explanation are left out and will be introduced latter in this section. Each block can be seen as a group of neurons, the basic unit of a neural network. In the most basic form of a neural network, each neuron receives as an input the output of all the neurons of the previous layer. Each neuron then linearly combines its input values, applies a non-linear activation function, and

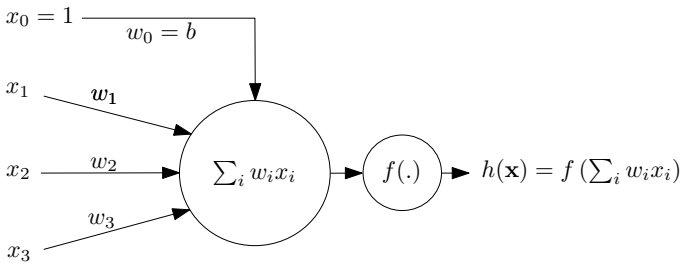


Figure 2.2: Representation of a single neuron. The input features x_1, x_2, and x_3 are linearly combined and weighted with the corresponding weights w_1, w_2, and w_3. The weight w_0 corresponds to the bias of the neuron and is added to the linear combination. The non-linear function $f(\cdot)$ is then applied to the output.

provides its output to the neurons in the following layer. As Figure 2.2 shows, the inputs on the left (x_1, x_2, and x_3) are linearly combined with the weights (w_1, w_2, and w_3). The non-linear function $f(\cdot)$, known as an activation function, is then applied to the result. Additionally, each neuron has a bias b that is added to the matrix multiplication; for the notation in this thesis, the bias term corresponds to weight w_0 and the respective input feature x_0 is assumed to be 1. Hence, the output of a single neuron becomes $h(\mathbf{x}) = f(\mathbf{w}^T\mathbf{x})$, where $\mathbf{w}^T = (w_0, w_1, w_2, w_3)$ are the weights (and bias) of the neuron and $\mathbf{x}^T = (x_0, x_1, x_2, x_3)$ are the features of the input vector (with $x_0 = 1$).

When the input is a vector of n features $\mathbf{x}^T = (x_0, x_1, \ldots, x_n)$, each layer of the neural network can be seen as a matrix multiplication where the input vector is multiplied by the weights of all the neurons in a layer $W = (\mathbf{w}_0, \mathbf{w}_1, \ldots, \mathbf{w}_d)$, a matrix of dimensions $n \times d$, where $\mathbf{w_i}^T = (w_0, w_1, \ldots, w_n)$ represents the vector of weights of the neuron i. This linearly projects the input to a d dimensional space, that corresponds to the number of neurons in a layer. Then, the element-wise non-linearity $f(\cdot)$ is applied resulting in the output $f(W^T\mathbf{x})$. Each layer often reduces the dimension of the previous layer output until the initial feature is mapped to the dimension corresponding to the output space (e.g. a class probability vector or a regression value). Note that, since compositions of linear functions, e.g. matrix multiplications, result in other linear functions, non-linear activations between layers are necessary to allow neural networks to express non-linear functions.

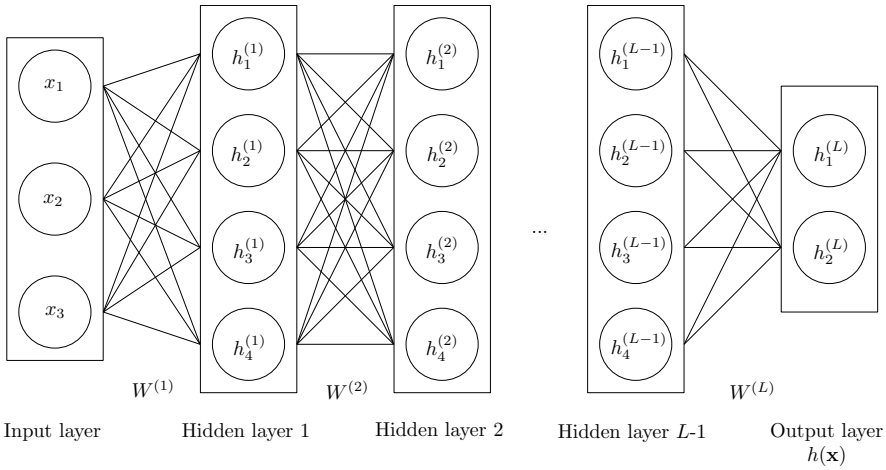


Figure 2.3: Multi-layer structure of a neural network with $L + 1$ layers: an input layer with 3 elements, $L - 1$ hidden layers of 4 dimensions each, and an output layer of 2 dimensions. The features h^{i-1} of each layer i result from the multiplication of the layer input features h^{i-1} by its weights W^i.

Figure 2.3 shows a common representation of a neural network. The first layer on the left constitutes the input layer: a vector of characteristics of the input samples (e.g. image pixels for image classification or word embeddings from a sentence for machine translation). The following layers comprise several neurons and are called hidden layers. The last layer on the right is the output layer that provides the prediction from the neural network for a particular input.

There are two main drawbacks when this structure is applied to images. First, the number of connections between layers from these models will increase rapidly with the increase of the characteristics in the input samples. For example, an image of 256×256 pixels with 3 channels for the colors (RGB) will need $256 \times 256 \times 3$ connections in the first layer (196 608 connections) for every neuron in the following layer. This is one of the first challenges that neural networks faced: scalability. Second, to consider each of the individual pixels from an image results in a model that does not take advantage of the possible local spatial structure from the data. For example, when a neural network is trained to classify images where the pixels are shuffled following a fixed random permutation, it performs as well as when it is trained with the original images. These two challenges motivated the development of convolutional neural networks (CNNs).

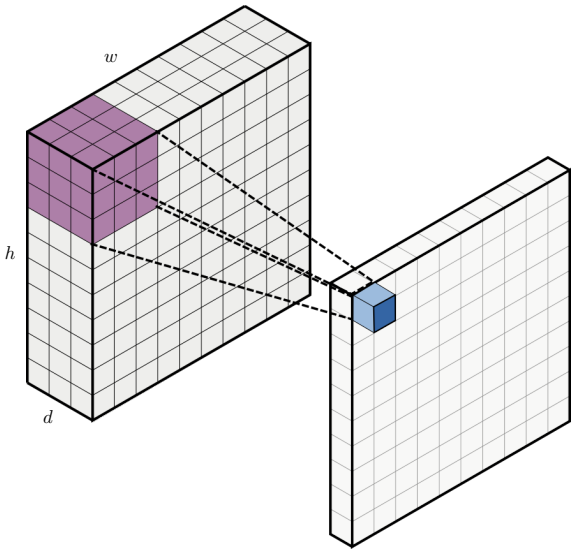


Figure 2.4: Convolution operation. In the input case, when dealing with images, the convolution is applied to a 3 dimension tensor, where the width (w) and height (h) correspond to the number of pixels of the input and the depth or channels (d) to the RGB color encoding. In this representation, the filter K is depicted in purple and has $3 \times 3 \times 3$ dimensions. Figure adapted from [118].

The main characteristic of CNNs is that they show each neuron a group of adjacent input features (a squared grid of pixels in visual tasks), rather than every characteristic of the input, and multiply them by a set of weights that are shared across all the input. This operation is called (discrete) convolution [55] (see Figure 2.4 for an illustration), and formally can be written as

$$S(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n), \tag{2.1}$$

where, in convolutional neural networks terminology, I is the input and K is called kernel or filter and corresponds to a set of weights associated to a neuron. Then m and n correspond to the dimensions of the input. It is often assumed that the values of K are zero everywhere except where it stores the values of the filters, which corresponds to the perceptive field of the neuron, i.e. the span of features that the neuron will take into account for each operation. Despite Eq. (2.1) being the formal definition of convolution, often in machine learning software libraries this is

implemented as the cross-correlation function

$$S(i,j) = \sum_m \sum_n I(i+m, j+n) K(m,j), \tag{2.2}$$

where the kernel is not flipped with respect to the image and maps more intuitively to the filtering concept in image processing. Note that in the applications considered in this thesis, there is no difference in the functionality of Eq. (2.1) and Eq. (2.2)

The cross-correlation in Eq. (2.2) is often referred to as convolution and is equivalent to the classic operation of image filtering, where a pattern described by a filter (constituted by the weights in K in this case) is multiplied by every position in the image. When the pattern from the filter correlates with a patch from the image, it results in a high value and activates the corresponding neuron. In other words, each neuron responds to a specific pattern in the image: the result of applying one neuron to an image would be a map that indicates where the pattern from one filter matches the image. This reduces the number of weights from the model. Additionally, convolutions allow working with variable input sizes and give CNNs a property called translation equivariance, by which spatial translations in the input produce equal spatial variations in the output.

The structure of CNNs

Each neuron activation is often considered as an output channel of a layer and all the activations constitute the input of the following layer. Each layer, then, receives as an input a tensor with spatial dimensions $h \times w$ and a depth or channel dimension d. In computer vision tasks, in the first layer, this corresponds to the raw image *pixels per row* $\times$ *pixels per column* $\times$ *RGB color encoding*. The basic block of a CNN consists of three parts, a convolutional layer, a non-linearity, and a pooling layer. The convolutional layer linearly transforms the input by applying the kernels introduced in Eq. (2.1), which in the image domain can be thought of as combinations of filtering operations including edge detectors and Gaussian blurs among others. A non-linearity often follows the convolution layer and allows the

stack of linear convolutions to represent non-linear functions. The pooling layers slowly reduce the spatial dimensions of the representations by replacing each element with summary statistics of its surrounding elements. This gives CNNs invariance to small translations on the images and is most useful when the final task depends on the presence of a feature in the input rather than its precise location. A common usage of pooling operations is to adapt CNNs to the fully connected nature of the classifiers in the final part of a model, where the input feature dimensions are fixed regardless of the input image dimensions. Note that sometimes pooling layers are substituted by larger strides in the convolution process (this corresponds to skipping input features in the spatial dimensions during the convolution operation), which also results in a reduction in the spatial dimension of the output.

Other elements often found embedded into CNNs are normalization layers that aim at stabilizing the training process by keeping the statistics of certain parts of the network normalized (zero mean and unit standard deviation): the statistics of the layer input [80], or the layer weights themselves [147]. The following section (2.1.2) introduces the functionality of normalization layers in more detail and explains the principles of training CNNs.

2.1.2 Training the model: back-propagation, gradient descent, and commonly used optimizers

The process by which the weights (or parameters) of the models are updated consists of a series of iterations where the samples from the training set are presented to the model and the respective outputs are evaluated on the given labels. The error of these predictions is evaluated through a cost or loss function and used to update the parameters of the model in the direction that minimizes the value of this function. The most commonly used loss function for image classification is the cross-entropy between the prediction and the target label,

$$\mathcal{L}(W) = -\frac{1}{n}\sum_{i=1}^{n} \mathbf{y}_i^T \log(h_W(\mathbf{x}_i)), \tag{2.3}$$

where n corresponds to the number of training samples and $\mathbf{y}_i \in \{1,0\}^c$ is the one-hot encoding label[1] for c classes corresponding to the input image $\mathbf{x}_i$, i.e. $\mathbf{y}_i$ is a vector with c dimensions, all zero except for the position corresponding to the index given by the ground truth label associated to the sample $\mathbf{x}_i$. $h_W(\mathbf{x}_i)$ correspond to the normalized output of the last layer of the model: a neural network with parameters W. The most widely used normalization function in image classification is called *softmax* and for the c dimensional output $\mathbf{z}$ of the last layer of the model, corresponds to

$$softmax(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{c} e^{z_j}}, \tag{2.4}$$

where z_i is the value in the i^{th} position of $\mathbf{z}$. In simple terms, the standard setup for classification maximizes the probability value assigned by the CNN to the class indicated by y_i.

The optimization of the loss function $\mathcal{L}(W)$ is often guided by gradient-based methods, which search the directions of the loss landscape defined by the negative gradients of $\mathcal{L}(W)$ with respect to the model parameters W. Concretely, variations of gradient descent (GD) are the main algorithms used for this purpose. Gradient descent starts with a random initialization of the weights W that define an initial point in the loss landscape. From there, gradient descent computes the derivative or gradient of the loss function with respect to the weights of the model and updates the parameters in the opposite (negative) direction of the gradient. This is often written as

$$W_{t+1} := W_t - \alpha_t \nabla_{W_t} \mathcal{L}(W_t), \tag{2.5}$$

where W_{t+1} are the weights of the model after the GD update, W_t the current weights of the model, $\mathcal{L}(W_t)$ the loss of the model with the current weights, and ∇_{W_t} denotes the gradient operation with respect to W_t. The parameter α_t defines the size of the steps that the optimization algorithm will take and is known as *learning rate*. It is one of the most important hyperparameters to train neural networks: a large value

[1] In machine learning, it is common to convert certain categorical data into vectors where each dimension corresponds to each category of the categorical distribution, the category of the given data point is encoded with the value 1, and the rest with 0

will make the optimization algorithm oscillate between points with high loss values and skip local optima – the optimizer might even diverge and move away from local minimum towards regions with higher values of the loss – and a low value will result in slow convergence of the optimization algorithm and could result in convergence to sub-optimal minimum. A common approach is to select a reasonably high value at the beginning of the training (typically 0.1) to allow the model to explore the loss landscape, and reduce it as the training goes on (typically by a factor of 10 at one third and two thirds of the training).

In a simplified manner, gradient descent computes the steepest direction in the loss landscape and does a step towards the direction that implies a fastest reduction of the loss value. Once the weights are updated accordingly, gradient descent takes another step following the same strategy. This process is iterated until convergence. The parameters are usually randomly initialized at the beginning of the training, and is important to note that, due to the nature of the non-convex loss function, the convergence point will change depending on the initialization.

Gradient descent becomes unpractical when the dataset size increases: for each step the model needs to process all the samples in the dataset. Additionally, given the redundancy in the datasets, computing each gradient descent step with all the samples makes it inefficient: most of the samples will have a similar contribution in each step. This drawback motivates the introduction of some of the variations of gradient descent. In particular, stochastic gradient descent (SGD) overcomes this drawback by approximating the loss gradient of the dataset with the gradient of the individual loss of each sample:

$$\ell_i(\mathbf{x}_i, \mathbf{y}_i, W) = -\mathbf{y}_i^T \log(h_W(\mathbf{x}_i)). \tag{2.6}$$

This, however, gives very noisy approximations of the gradients and forces the model to process each sample individually, updating its parameters for every sample. The most common alternative of gradient descent is mini-batch SGD [41, 22] and lies in between GD and SGD: parameter updates are computed from batches of samples.

This alternative is often referred to as SGD and estimates the real gradient of the loss function as the average gradient of a randomly selected subset of the samples. The main advantage of this optimization algorithm comes from the computational point of view. Since current GPUs are designed to optimize parallelized computations, the gradient estimation of a batch of samples has the same computational cost as the estimation of the gradient estimation of a single sample. Hence, the number of parameter updates can be drastically reduced. Mini-batch SGD often considers from 10 to 200 samples at every iteration, this also provides a benefit from the optimization point of view: averaged gradients are less noisy and provide certain robustness to corruptions or outliers.

SGD can be applied to all kind of models, such as logistic regression, linear regression, or neural networks. In the case where the prediction function of the model is a neural network, backpropagation provides an efficient method for calculating the gradient of the loss. The underlying principle of backpropagation relies on the chain rule to compute the gradients of the errors from different layers with respect to their weights. This gradient can be interpreted as how sensitive is the error of one layer to changes in the weights of that layer and, more importantly, gives the local direction of steepest ascend. As we are interested in the derivative of the loss function with respect to the weights, in each layer, the chain rule

$$\frac{\partial \mathcal{L}(W)}{\partial W} = \frac{\partial \mathcal{L}(W)}{\partial \mathbf{o}} \times \frac{\partial \mathbf{o}}{\partial W}, \tag{2.7}$$

decomposes this derivative into two terms: the derivative of the loss function $\mathcal{L}(W)$ with respect to the output of the preceding layer $\mathbf{o}$, and the derivative of the output of the previous layer with respect to the weights of that layer W. For the previous layer, the gradients are computed similarly using the loss function associated to that layer. This way, as the name backpropagation suggests, the error computed in the last layer is propagated backwards through the network to compute the gradients for each layer.

Despite being the algorithm behind most neural network applications, stochastic

gradient descent can be sometimes a slow optimization strategy: the gradients from different batches have a stochastic component that increases the exploration of the loss landscape through noisy updates of the weights, but discourages a straightforward convergence towards low loss regions. Particularly, in the presence of ravines, where the surface curves much more in one direction than in others [171], SGD updates oscillate and become noisier. To accelerate the convergence of SGD, momentum [145] is often applied: an exponentially decaying average of past gradients is used to update the weights of the model. Formally, the variable $\mathbf{v}$ accumulates the negative gradients of the loss with respect to the parameters of the model and accounts for the update of the model as

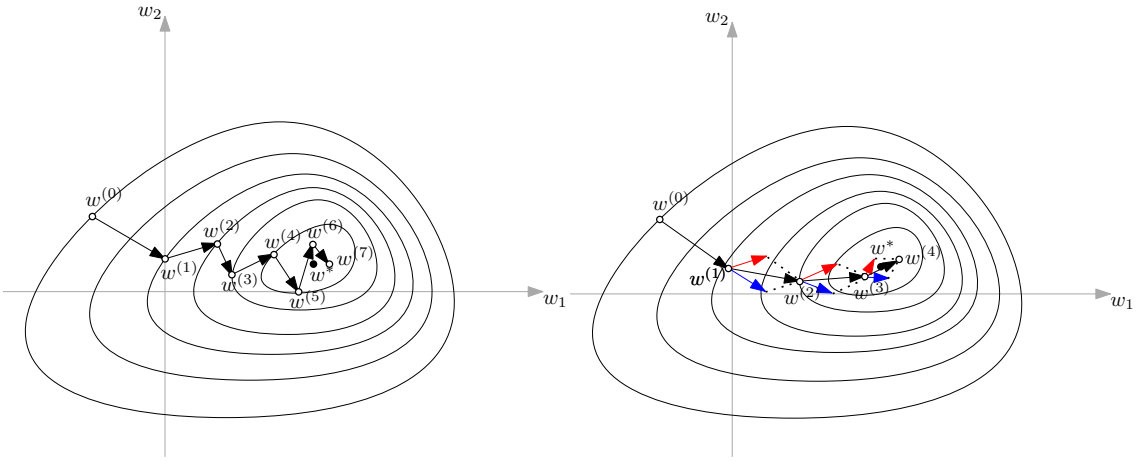
$$\mathbf{v}_{t+1} := m\mathbf{v}_t - \alpha_t \nabla_W \mathcal{L}(W_t) \tag{2.8}$$

$$W_{t+1} := W_t - \mathbf{v}_{t+1}, \tag{2.9}$$

where α_t is the learning rate and $m \in [0, 1)$ is the momentum parameter that determines how fast the contribution of previous gradients decay.

More intuitively, momentum is decreasing the effect of a particular update considering the *momentum* with which the loss landscape is being explored and reducing oscillations on the way to the local optimum. Figure 2.5 (b) illustrates the difference between SGD and SGD with momentum.

Further improvements in the optimization algorithm aim at considering different step sizes (i.e. learning rates) for different directions in the loss landscape. This is motivated by the high dimensionality of the loss landscape when training neural networks. Since higher dimensional spaces present a higher number of saddle points (points where the loss increases in some directions and decrease in others) than local optima (points where the loss increases in all directions) higher learning rates in directions in which the gradients are smaller help the optimization algorithm to escape those saddle points. Examples of the most used algorithms with adaptive learning rates are AdaGrad [48], RMSProp, and Adam [96]. However, for simplicity



(a) Stochastic gradient descent (SGD) (b) SGD with momentum

Figure 2.5: Momentum effect on stochastic gradient descent. Black arrows indicate the SGD step taken, red arrows indicate the step computed from the current gradient of the loss, and blue arrows indicate the accumulated previous gradients. Figure adapted from [118].

and to avoid additional hyperparameters, the experiments on this thesis use SGD with momentum.

It is important to remember that the central challenge of neural networks – and in machine learning approaches in general – is to perform well on unseen examples, i.e. samples that are not in the training set. This ability is known as generalization and is evaluated with the generalization error: an error measure over the testing set (unseen samples). The two main challenges when approaching generalization are underfitting and overfitting. The former corresponds to the lack of enough flexibility or capacity in the model to perform well on either the testing or the training set. This is usually not a problem in neural networks given their overparametrized nature: common architectures have enough parameters to memorize the training set and achieve zero loss (note that this is true even for random samples [216]). The latter, however, refers to improvements of performance in the training set that do not corresponds to improvements on the testing set. This difference between test and training errors is called the generalization gap and is a direct measure of the overfitting of a model. This challenge is not as straightforward as the lack of capacity in the model and is strongly affected by the overparametrization of the model.

Regularization techniques play a crucial role to mitigate the effects of this intricate

relationship between overparametrization and generalization. Current practices in the field ensure, firstly, enough capacity in the model, and then apply regularization techniques to bias the model towards certain solutions that are more likely to be useful in the testing set (i.e. more likely to generalize). Regularization techniques restrict the capacity of the model and force it to learn certain patterns. Widely used regularization techniques encourage the parameters of the model to remain close to zero by including a penalty that keeps the parameters Euclidean norm small (L2 weight regularization [99]), drop random parameters at each iteration to reduce the co-dependence between them and leverage the ensemble effect during testing (dropout [167]), or introduce invariance to particular functions of the input space by artificially altering the inputs with certain transformations that conserve the semantics (i.e. data augmentation [160]). In other words, regularization techniques increase the difficulty of the training set and encourage robustness to some variations during training to avoid overfitting.

Additional limitations of neural networks concern the preparation of the input data and the effect this has during training. A common practice in machine learning and particularly when training neural networks is to standardize the input data: subtracting the mean and dividing by the standard deviation. This centers the data in an approximately spherical region (with unity radius) of the space around zero. By having all the data normalized, the network parameters do not need to compensate for differences in the ranges of different feature dimensions. This results in initial weights being closer to the final solution. While this is done in the input, other techniques aim at maintaining the statistics of the internal representations of the neural network also near zero mean and unity standard deviation. These are often seen as normalization layers that keep the activations of previous layers standardized using the statistics of a mini-batch (batch normalization [80]), normalize the weights in each layer (weight standardization [147]), or normalize the output features of each layer grouped by channels (group normalization [199]). While understating the role of these layers is still an active research topic, they are often used to stabilize the

training and accelerate convergence.

2.1.3 Widely used model architectures

Initial implementations of convolutional neural networks process the input data from layer to layer until reaching the output space that, in the classification task, is normalized to obtain a probability distribution over the classes in the dataset. AlexNet [1] is an early example of this architecture that has 5 convolutional layers followed by 2 fully connected layers. Soon after this contribution to the ImageNet challenge, the number of layers used to design CNNs increased, as more layers meant more parameters and thus, more representation power. This gave place to VGG architectures [162] as *deeper* and refined versions of AlexNet.

Despite the competitive performance of larger architectures such as VGG16 and VGG19 – which increased the number of layers up to 16 and 19 respectively – training these deeper models became increasingly challenging: larger number of layers presented a problem for backpropagation, as the gradients become smaller as they reach deeper layers. Since the weights of the network are encouraged to be small, this results in the gradients being multiplied by values smaller than one, which reduces their value to near zero after several multiplications. This is known as the vanishing gradient problem. Consequently, ResNet [72] architectures include a skip connection between blocks of convolution layers to avoid this problem. Skip connections forward the input of each block of typically 4 layers to its output; the output of each block, then, is the addition of the input and the output features of the block. This encourages stability between blocks and allows the gradients to flow backwards during backpropagation and reach deeper layers by avoiding multiplications that reduce their magnitude. Common ResNet architectures like ResNet18, ResNet50, and ResNet101, where the numbers indicate the number of layers, have become the default architectures in several computer vision tasks.

Later innovations in CNNs included skip connections between all the layers in the network to facilitate feature reuse and gradient propagation, resulting in the

DenseNet architecture [78]. MobileNet [76] substitutes the convolutional layers for depth-wise separable convolutions to reduce the number of parameters in the model. This architecture is designed for optimal performance on mobile devices and includes two hyperparameters to allow for a trade-off between latency and accuracy. More recently, Mingxing Tan and Quoc Le [174] proposed EfficientNet as a new architecture found through model architecture search strategies [229, 173]. Additionally, they optimize the existing ResNet and MobileNet architectures with respect to the depth, width, and resolution they use. Note that since the success of AlexNet, the CNN landscape has been growing with an almost overwhelming variety of architectures. This thesis does not aim at exploring the different architectures available and, for simplicity, uses standard variations of ResNet-type and VGG-type networks.

2.2 Literature review: approaches to constrained training setups

In this section, we introduce the relevant research on neural network approaches to computer vision under limitations of label availability and computational constraints. In particular, this section addresses the literature on unsupervised and self-supervised learning, semi-supervised learning, label noise, and budgeted training.

2.2.1 Unsupervised and self-supervised learning

The initial success of CNNs in unsupervised learning for computer vision was driven by self-supervised approaches that obtained the training labels directly from the data. These approaches leveraged some visual property of the images to provide a signal to train the model. Concretely, they aimed at solving proxy tasks such as jigsaw puzzles, predicting the color of an image, or identifying rotations [132, 221, 53]. More recently, self-supervised approaches have made a shift towards metric learning inspired objectives, in particular to contrastive representation learning frameworks [93, 94, 180] (often referred to as contrastive learning), where similar

samples are pulled together in the representation space while dissimilar are pushed apart [29, 70, 201, 31]. When applied to unsupervised learning, contrastive learning pulls together two or more random transformations of one image and pushes apart all the other images.

Most self-supervised approaches based on proxy tasks are inspired by intuitive ideas of how humans learn visual patterns. For instance, Doersch *et al.* [45] and Noroozi *et al.* [132] build a task that encourages the network to understand the spatial location of certain features from the image. Doersch *et al.* [45] use as an input two patches from the same image and train a CNN to predict the location of one patch with respect to the other. Similarly, Noroozi *et al.* [132] train a model to solve Jigsaw puzzles: a 3×3 grid in the image is reorganized according to a random permutation from a list and the network is trained to predict the index of the given permutation in the list, i.e. the list index acts as a class. These works suggest that this scenario encourages the model to learn a feature mapping of different object parts and their spatial arrangement with respect to the whole object. Another commonly used pretext task is colorization. Zhang *et al.* [221] propose an early approach for this task that predicts the channels *ab* given only, as an input, the information from the L channel in the *Lab* color space. In this scenario, the model is not predicting classes anymore, now the predictions are pixel values that account for the color of the image. Guadarrama *et al.* [60] bring this a step further and proposes a recursive colorization to generate a high-resolution image colorization.

While these approaches require considerable modifications to the training process, Gidaris *et al.* [53] propose a very simple and effective approach: they train a network to identify which rotation was applied to an image; in this setup, the model predicts the rotation applied to an image from fixed given degrees of rotation (e.g. {0, 90, 180, 270}) as if these were classes. The intuition behind this approach is that rotations force the network to recognize object semantics in order to be able to determine their orientation.

In a more transformation-agnostic approach, Dosovitskiy *et al.* [47] propose

Exemplar-CNN to train a CNN to identify augmented patches from one image as examples belonging to that specific image. These augmented patches are crops of an image that undergo several random visual transformations (translation, illumination changes, rotations, scale changes, and horizontal flips) and aim at reproducing realistic views of the given images. As a result, a group of transformed patches from the same image form a surrogate class per image. This method is of particular interest because it is not limited to a given pre-designed proxy task, rather it aims at training a model that is invariant to generic transformations from the input space.

Exemplar-CNN, however, presents three main weaknesses. First, images from the same semantic class end up in different classes (see Figure 2.6), generating different surrogate classes and creating collisions and misguiding the training . As a result the performance of this approach degrades as the number of available unlabeled samples increases. To alleviate this effect, Dosovitskiy *et al.* [47] and the work [9] presented in Chapter 3 introduce clustering methods that allow them to increase the amount of unlabeled data used for training. Secondly, the method ignores intra-class and inter-class relationships: relationships between elements from the same semantic class and between elements from different classes. This is studied by Bautista *et al.* [17], where they design the training mini-batches to contain samples from different classes. Third, the large number of classes of the resulting dataset makes the *softmax* normalization ineffective, which provides a weaker guide for the parameters updates. Doersch and Zisserman [46] use a triplet loss approach to address this last challenge, which points towards the recent shift from visual proxy tasks to metric learning approaches (further discussed in Chapter 3).

Recent approaches include random artificial variations in the images, and shifting of the training paradigm from classification to metric learning, or more concretely, to contrastive learning. In this scenario, the underlying idea is to minimize the distance between similar images in the feature space while maximizing the distance between dissimilar images. In the case of unsupervised learning, the similar images are patches of the same image that undergo different random transformations. Oord *et*

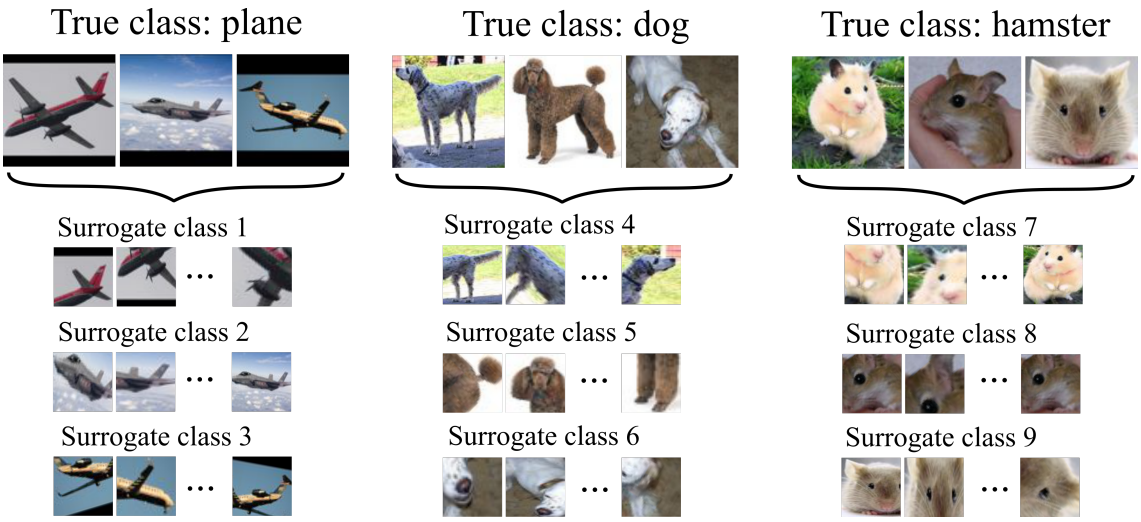


Figure 2.6: Collisions between surrogate classes. Several patches are extracted and randomly transformed to create a surrogate class per image, regardless of the true class. This process results in a dataset with different classes coming from the same true (unknown) class.

al. [135] and Wu *et al.* [200] first used contrastive losses for self-supervised visual representation learning and proposed to adapt noise-contrastive estimation (NCE) as a strategy to approximate a normalization of the predictions – often used in language tasks to learn word embeddings [120, 125]. The basic idea of these methods is to substitute the linear layer and *softmax* normalization that projects the predictions to probabilities in the class space, which in this case is constituted by as many classes as there are samples in the dataset, with a non-parametric classifier, where the probability assigned to each sample comes directly from computing its distance to different transformations of this sample and normalizing over the sum of distances to all the other samples in the dataset. For large datasets, this is infeasible, hence NCE approximates the normalization over a relatively small subset of samples. This translates into a loss that minimizes the distance between the given sample and a positive (itself with a different augmentation) and maximizes the distance of the given sample with a set of negative samples (e.g. the other samples in the mini-batch or the samples from a memory bank).

Soon after these initial attempts, other approaches leveraged this idea and proposed methods to learn representations contrasting distances in the feature

space [70, 29, 181, 89, 102, 27, 59, 32]. In particular, SimCLR [29] is a method that considers as positive the second view of each image, i.e. the same image under a different transformation, and as negatives all the samples in the mini-batch. This approach introduces stronger data augmentation to obtain more challenging positives, a large batch size to increase the number of negatives, and a non-linear layer in the projection module of the network.MoCo [70] and MoCo-v2 [31] introduce a memory bank and a momentum network to the pipeline, which allows to compute better representations and to access a higher number of negatives per sample. Tian *et al.* [181] explore the influence of the view generated for contrastive learning and propose InfoMin as an approach to reduce the mutual information between views and maintain task-relevant information. Caron *et al.* [27] replace the feature comparisons by enforcing consistency between clusters assignments of different views from the same image. Kalantidis *et al.* [89] and Lee *et al.* [102] introduce interpolation training to obtain harder negatives and positives, which plays an important regularization role in boosting performance.

Slightly different, Grill *et al.* [59] propose BYOL, which does not contrast representations between samples, ignores the negatives, and aims at predicting one view from another. Chen and He [32] further simplify this setup and proposes a framework in which all contrastive approaches can be seen as siamese networks [24]. They show that by stopping the gradient propagation of one of the views of the mini-batch they avoid collapsed solutions where the model learns to predict constant features and allows the model to train without memory banks, large mini-batches, or momentum networks.

Despite the good performance of these methods, they still present several inconveniences when training. The number of negatives needs to be considerably large, which is often solved with a large batch size [29] or with a memory bank [70] that is updated through training. Both alternatives are computationally demanding. The second drawback is the typically large number of iterations required to train these methods, that due to the weak guidance from a single positive for every update, the

training needs to be extensively long (as seen in supervised contrastive learning [94], where the availability of several positives per image greatly accelerates convergence). This is aggravated by the heavy data augmentations applied to the positives and negatives [102, 29, 31], which make it even harder for the model to learn representations. Chapter 3 discusses these challenges in the light of Exemplar-CNN [47] as an approach that contrasts samples with themselves and an early attempt at contrastive learning for unsupervised learning.

2.2.2 Semi-supervised learning

Semi-supervised approaches propagate label information from a small subset of labeled data to a larger subset of unlabeled samples and jointly exploit both subsets to learn discriminative representations. Additionally, semi-supervised learning plays an essential role in the application of self-supervised representations to downstream tasks. Hence, the recent trend in self-supervised works of evaluating the learned representations in semi-supervised benchmarks [29, 59, 27].

For clarity, in this thesis, approaches to semi-supervised learning are divided into two categories: pseudo-labeling and consistency regularization. Approaches in the former consider the given labels as the training objective for the labeled set and estimate pseudo-labels from the predictions of the model as training objectives for the unlabeled set [101, 81, 11]. The latter, however, does not attempt to generate labels for the unlabeled samples, but to enforce consistency across the predictions of a sample under different perturbations (data augmentation, dropout, or other perturbations) [155, 176, 19]. In the presence of scarce annotated data, these approaches become vulnerable to confirmation bias [176, 106, 11]. While in humans, this bias manifests as a tendency of searching for and focusing on the information that confirms existing beliefs [143], in neural networks this bias manifests as a propagation of errors through the unlabeled set and as a tendency of the model to resist new changes [106]. Despite being present in pseudo-label and in consistency regularization, confirmation bias seems to be more damaging to the former. This is

further explored in Chapter 4.

Consistency regularization was used by Sajjadi *et al.* [155] by combining perturbations from randomized data augmentation, dropout, and random max-pooling and forcing the predictions of the model to be consistent. Similarly, *Temporal Ensembles* [100] extend these perturbations in time and encourages model predictions to be similar to a running average of the predictions across epochs, i.e. as a temporal ensemble. These predictions contain the additional corruptions coming from different update states of the network. In *Mean Teacher* [176] the temporal ensemble paradigm is interpreted as a teacher-student scenario where the main network behaves as the student by encouraging consistency of its predictions with the predictions provided by the teacher. The teacher, then, is defined as an exponential moving average of the weights of the student network. The update speed of this moving average depends on a hyperparameter and controls the confirmation bias of the model. Li *et al.* [106] extend this framework by weighing the unlabeled samples and giving higher importance to those with low uncertainty, defined as the entropy of the predictions for each sample. Miyato *et al.* [124] use virtual adversarial training to corrupt the samples with adversarial noise and later impose consistency across predictions. More recently, Luo *et al.* [113] propose to extend the consistency regularization term by introducing a graph to consider the connections between samples in the prediction space and enforce similarity between neighboring nodes and dissimilarity for non-neighboring nodes. This can be combined with the methods proposed by Tarvainen and Valpola [176] (*Mean Teacher*) and Miyato *et al.* [124] to boost their performance.

Verma *et al.* [188] propose interpolation consistency training as a method inspired by *mixup* [218] data augmentation (further explained in Subsection 2.2.5), that imposes consistency between the interpolated predictions of unlabeled samples and the predictions of interpolated unlabeled samples. Similarly, Berthelot *et al.* [19] propose MixMatch, that includes *mixup* to combine labeled and unlabeled samples, and applies consistency regularization between predictions and guessed labels obtained from averaging predictions under several augmentations while applying

a sharpening function to encourage low-entropy guesses. Both Verma *et al.* [188] and Berthelot *et al.* [19] adopt *Mean Teacher* [176] to estimate better targets for consistency regularization.

Qiao *et al.* [146] propose Deep Co-training as a method that simultaneously trains two (or more) networks, encouraging agreement on their predictions (consistency regularization) and disagreement on their errors. Those predictions that change when exposed to adversarial attacks are considered errors: this forces different networks to learn complementary representations for the same samples. Chen *et al.* [33] introduce a memory that store previous representations to be used to measure the consistency with current predictions. Additionally, they introduce a term to reduce the contribution of the consistency term for uncertain samples.

Pseudo-labeling methods have been less successful in learning representations. By using model predictions directly as labels, these methods spread possible errors in the predictions through the unlabeled set which reinforce learning corrupted representations. This phenomena, i.e. confirmation bias, has kept approaches to semi-supervised learning away from pseudo-labeling strategies.

An early attempt at pseudo-labeling [101] used the predictions of the model as labels for a fine-tuning stage after pre-training the model on the labeled set of samples. More recently, Shi *et al.* [159] use the predictions of the model as hard labels for the unlabeled set (i.e. the class predicted with the highest probability becomes the label). The authors introduce a per-sample weight that reduces the contribution to the loss of those samples whose k-nearest neighbors are furthest in the feature space. They include two additional terms on the loss function, one to encourage intra-class compactness and inter-class separation, and the other to encourage consistency between samples with different perturbations. They also report improved results when combined with *Mean Teacher* [176]. Recently, Iscen *et al.* [81] exploit graph-based label propagation in the pseudo-labeling paradigm: the labels of the clean set are propagated to the whole dataset to refine the pseudo-labels for the unlabeled set. Concretely, the method alternates between a stage where the

network trains on the labeled and pseudo-labeled data, and a stage in which the representations of the network are used to build a nearest neighbor graph where label propagation [227] is applied to refine hard pseudo-labels. They further add an uncertainty score for every sample (based on *softmax* prediction entropy) and class (based on class population) to deal, respectively, with the unequal confidence in network predictions and class imbalance.

Rebuffi *et al.* [149] combine the pseudo-labeling paradigm with an alternating optimization method to avoid propagating errors through the dataset (confirmation bias): In the first stage, the classifier is trained on the clean set of data and produce pseudo-labels for the unlabeled set that are then used in the second stage to fine-tune the full network on the unlabeled set. Additionally, similarly to the study by Zhai *et al.* [215], they include self-supervised learning in the pipeline, particularly as a warm-up to initialize the network. This was later adopted by several other approaches [20, 19, 165] and has shown to help to stabilize the training in cases where the number of labeled samples is very low.

The work proposed in Chapter 4 explores the main weakness of pseudo-labeling approaches, i.e. confirmation bias, and leverages simple regularization techniques that address it: a minimum number of labeled samples per batch and *mixup* data augmentation. These regularization techniques allow the network to generate reliable soft pseudo-labels and to learn robust representations, even when the number of labeled samples is extremely low (down to 0.5% of labeled samples). Additionally, the loss function proposed leverages existing regularization terms that enforce an equal prediction of the samples per mini-batch and encourage low-entropy pseudo-labels.

The distinction between pseudo-labeling and consistency regularization approaches is very useful from a technical point of view, but it has become less evident in recent approaches. As noted by Rebuffi *et al.* [149], a pseudo-labeling strategy across epochs has a consistency regularization effect where predictions for one sample are encouraged to be similar from epoch to epoch. Recent research in semi-supervised learning propose holistic approaches that combine several of the techniques used in previous

works [19, 20, 165] and in some cases make the distinction between consistency regularization and pseudo-labeling impossible. Initially, MixMatch [19] combines, *mixup* as an interpolation based data augmentation, standard data augmentation to generate label guesses for the unlabeled set, entropy minimization to sharpen those guesses, and a teacher model as in *Mean Teacher* [176] to improve the quality of the guesses used in a consistency regularization term in the loss function. Soon after that, Berthelot *et al.* [20] proposed ReMixMatch, that improves on MixMatch by introducing stronger data augmentation and a self-supervised term in the loss to increase stability during training when the number of labeled samples decreases. Additionally, they substitute the consistency regularization term in the loss function (the mean square error is used often for this term) by the cross-entropy between a guessed label and the prediction of the model, as is done in pseudo-labeling approaches. This relationship between consistency regularization and pseudo-labeling is further discussed in Chapter 4. To further improve on this, Sohn *et al.* [165] propose FixMatch that includes a sample selection step where only the high-confidence predictions are used.

To increase the efficiency of semi-supervised learning approaches when extremely scarce annotations are available, Albert *et al.*[4] propose to learn features in a self-supervised manner and leverage the relationships between samples in the feature space to extend the labeled set of samples. This allows semi-supervised training in challenging scenarios where as few as one single label per class is available.

While in this thesis we assume that unlabeled samples belong to the distribution of classes in the labeled set, recent semi-supervised learning approaches explore more realistic scenarios where the unlabeled samples do not necessarily belong to the label distribution [34, 61, 228]. The research in this direction is still very limited, but tackles one of the main limitations of current semi-supervised methods. The second assumption considered in this thesis is that the labeled samples are balanced across all the classes. Several approaches already consider cases where this is not true [209, 109, 195]. The implications of these two lines of research and their relation with

existing approaches are further discussed in Chapter 7.

2.2.3 Label noise learning

Neural networks learn corrupt representations when trained under label noise conditions: their expressive power leads them to memorize the corruptions in the labels [216, 14], reducing their generalization abilities to unseen samples. This is a common scenario, especially in large datasets where the labeling process is automated and the labels come from existing information such as user tags, text, or web searches [105, 204]. Recently, increased attention in this field [51, 5, 65, 166] has resulted in new benchmarks [85, 105], approaches [10, 137], and insights on the training of neural networks under label noise [136, 64].

To facilitate the exploration of label noise approaches, it is common to introduce label corruptions in classification benchmarks to create controlled scenarios with known noise distributions and where noisy samples are identified. Label noise distributions are often grouped into two categories: closed and open set distributions. These are also known as in-distribution and out-of-distribution label noise respectively: in the former, the true labels of noisy samples belong to the labels inside the expected distribution of classes, while in the latter they often do not. An additional assumption when building synthetically corrupted datasets is the distribution of the corruptions across the classes: uniformly or non-uniformly distributed. In uniform noise distributions, the labels are swapped uniformly between all the classes, and in non-uniform noise distributions, the corruptions follow a certain class-dependent distribution where some classes might swap labels more often than others. As noted in Ortego *et al.* [136] real-world noisy datasets are likely to contain both types of noise distributions.

The two setups most commonly studied in the label noise literature are uniform and non-uniform noise distributions under the closed set scenario assumption. The motivation behind this is the lack of datasets with controlled out-of-distribution noise. Recently Jiang *et al.* [85] released a new dataset where the noisy samples are

obtained from web searches: this allows for the selection of the level of noise and evaluation of the correct identification of noisy samples. Similarly Ortego *et al.* [136] proposed to use a down-sampled version of ImageNet to synthetically simulate these distributions: the 1 000 classes in ImageNet are divided into in-distribution and out-of-distribution, and the corruptions are introduced following a class-dependent distribution given by several pre-trained networks. These works aim at providing controlled versions of realistic datasets such as WebVision [105] or Clothing1M [204], that are extremely large, costly to work with, and where the noisy samples are not identified (which makes the understanding of the methods impossible).

Approaches to label noise follow four main different strategies: detect noisy samples and discard their labels [175, 213, 44, 95, 136] to, then, apply a semi-supervised algorithm; directly discard the samples or reduce their contribution to training [86, 62, 194, 85, 130, 64, 214, 196, 151]; correct the per-sample loss values or labels observed during training [150, 66, 10, 110, 141, 73, 202, 210, 187, 204]; or introduce a robust loss function to train in the presence of label noise [218, 103, 68, 177, 223, 207, 114, 186]. Figure 2.7 provides an overview of this literature review and groups the different methods under their corresponding category.

A conservative approach to label noise is to discard the labels associated to noisy samples but still leverage the images for representation learning. By discarding the noisy labels, these approaches avoid overfitting to the noise distribution of the dataset [175, 213, 44, 95, 136]. In this direction, Tanaka *et al.* [175] and Yi and Wu [213] demonstrate that, by replacing clean and noisy labels with the predictions of the model or with learned label probability distributions, they are able to mitigate the effect of label noise. Concretely, after a pre-training period, all the labels of the dataset are substituted by the predictions of the model; this aims at extrapolating the features learned during the initial stages of the training where the model is less prone to fit the noise (due to a larger learning rate). Intuitively, this leverages the observation by Arpit *et al.* [14] and Zhang *et al.* [216], of clean samples being learned earlier than noisy samples, to replace the potentially corrupted labels before

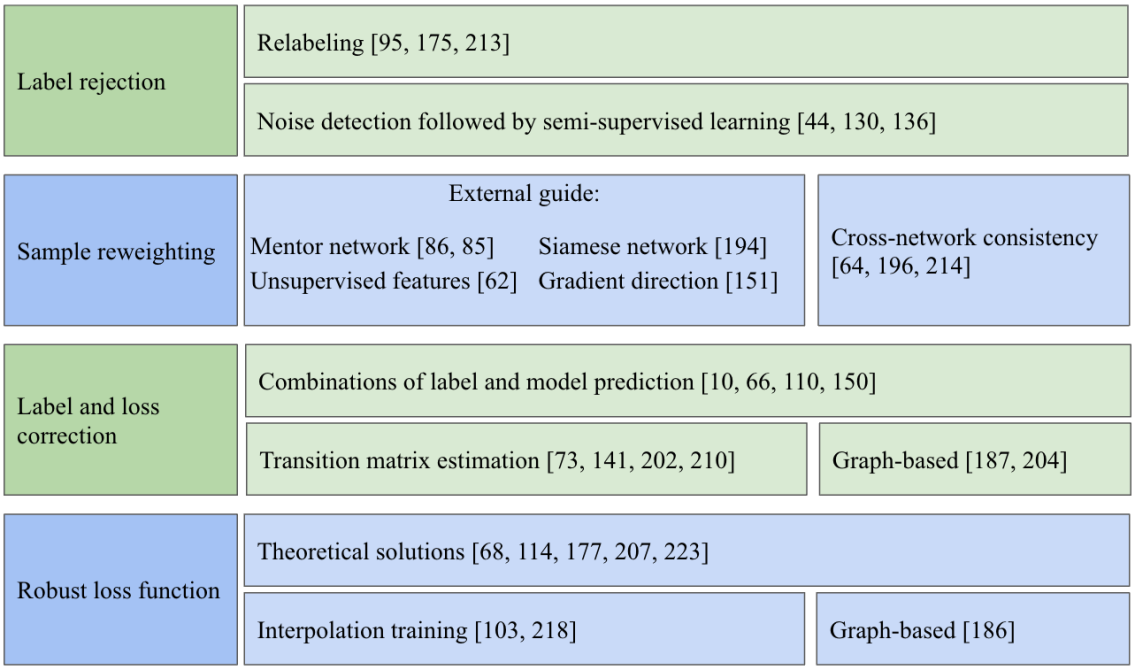


Figure 2.7: Compilation of label noise approaches.

the model overfits them. Kim *et al.* [95] introduces negative learning as an indirect training method to reduce the effect of noisy labels in training: samples are randomly replaced and contribute negatively to the loss function. In a similar direction, several works [44, 130, 136] introduce a noise detection step to identify the noisy samples and discard the associated labels. Then, semi-supervised learning approaches are applied to learn from both the clean and noisy (now unlabeled) samples.

Other approaches also try to identify the noisy samples, but with a different objective: remove them or reduce the influence they have on the training loss function [86, 62, 194, 85, 64, 214, 196, 151]. Particularly, Jiang *et al.* [86] include an additional model to the training, a mentor network (which was later combined with interpolation training Jiang *et al.* [85], i.e. *mixup*), to learn a curriculum from the data and score the probability of the samples of being clean. This curriculum is then used to present to the main model only the samples with a higher probability of being clean. Similarly, Guo *et al.* [62] design a learning curriculum through an unsupervised measure of the complexity of the data and use that curriculum to introduce the noisy samples gradually during training. This reduces their impact

on the representation learning and leverages their regularization effect. Wang *et al.* [194] weight the noisy samples to reduce their influence during training, and introduces a metric learning perspective to train the model to place far away in the feature space those samples identified as noisy. Some of these approaches include several networks to improve label noise detection, e.g. Nguyen *et al.* [130] iteratively clean the training set by leveraging the prediction agreements of ensemble networks. Similarly, cross-networks loss values and prediction disagreements have shown be useful to select which samples to backpropagate from [64, 214]. As an improvement to this, Wei *et al.* [196] include a consistency regularization term to force similarity of the features between both networks to further exploit the samples considered as noisy and discarded for the backpropagation associated to the labels.

Loss and label correction approaches to label noise often dispense with the noise detection stage and directly modify the loss values or the labels used during training to reduce the effect of label noise [150, 66, 10, 110, 141, 73, 202, 210, 187, 204]. Reed *et al.* [150] modify the loss function to reduce the impact of noisy labels; effectively the labels become convex combinations of the observed and the predicted label. This is extended by using prototypes as class estimations [66], and combination of labels and predictions dynamically weighted based on the clean-noisy per-sample probability given by an unsupervised modeling of the noise [10] (Chapter 5 expands on this work). Similarly, Liu *et al.* [110] introduce a term on the loss function to maximize the inner product between the model prediction and the targets to prevent memorization of noisy samples at later stages of the training. In a slightly different direction [141, 73, 202, 210, 187, 204] aim at estimating the noise rate in the dataset. In particular [141, 202, 210], estimate a transition matrix T that specifies the per-class probability of labels being flipped from one class to another. Patrini *et al.* [141] propose to correct the loss function by multiplying T by the *softmax* probabilities predicted by the model to correct the model predictions. Given the challenging nature of estimating this matrix, Yao *et al.* [210] propose a Bayesian nonparametric approach to estimate the transition probabilities and deduce a label

regression method to iteratively infer the labels to train the classifier and model the noise. Also, Xia *et al.* [202] exploit certain clean detected samples (highly reliable) as anchor points to estimate the transition matrix.

Approaches that explore robust loss functions [68, 223, 207, 114] provide alternatives to the standard cross-entropy training and explore a more theoretical approach to mitigate label noise memorization and encourage representation learning. Zhang *et al.* [223] present a set of noise-robust loss functions that exploit the mean absolute error in combination with the standard categorical cross-entropy loss. Xu *et al.* [207] propose an information-theoretic loss function based on a generalized version of the mutual information that is provably insensitive to noise patterns and corruption levels. Similarly, Ma *et al.* [114] theoretically show that current loss functions could be made robust to noisy labels with a simple normalization, but that this is not enough to train accurate deep neural networks; they propose a combination of two losses that mutually boost each other. Harutyunyan *et al.* [68] include a term on the loss consisting of the Shannon mutual information between the weights of the network and the vector, and show that minimizing this term corresponds with reductions in memorization of label noise and better generalization. More generically, Zhang *et al.* [218] propose *mixup*, an interpolation training strategy that reduces memorization in neural networks and increases robustness in the presence of noisy labels. This strategy has been extensively adopted in the label noise literature [136, 85, 150, 10, 103].

Other approaches propose mixed strategies [110, 103, 177] and achieve remarkable results. DivideMix [103] for instance, leverages cross-network agreements, semi-supervised learning, and interpolation training to train a model and at the same time refine the labels detected as clean. The approach proposed by Liu *et al.* [110] includes regularization terms that leverage the features learned during initial stages; the authors also use network ensembles and weight averaging, to significantly boost performance. Slightly differently, Thulasidasan *et al.* [177] propose a method that includes an additional class to the training and uses a penalty in the loss function to

encourage the network to predict noisy samples under that additional class.

In parallel to these approaches, some works relax the noise corruption assumption and rely on a clean set to enhance some aspects of the training [64, 73, 187, 204, 186]. Han *et al.* [64] consider the validation set as a clean set available during training and uses the directions of gradients on the training set compared to those on the validation set (clean set) to weight the influence of the samples in the loss function. Hendrycks *et al.* [73] leverage a small clean set to estimate a better noise transition matrix T to correct the predicted class probabilities. Veit *et al.* [187] propose to use the clean set to learn a mapping between clean and noisy annotations, and argue that this mapping learns noise patterns and captures the noise structure in the label space. Xiao *et al.* [204] propose a probabilistic model to describe the cause of noisy labels in real-world data and integrate this model with a CNN to improve training robustness under label noise. Finally, Vahdat [186] propose to use a conditional random field to represent the relationships between clean and noisy labels and embed it into the model to make the training robust against label noise.

2.2.4 Budgeted training

The rapid increase in training times given by larger datasets and deeper models provides a strong motivation to explore alternatives to reduce the computational costs of training deep neural networks. Some of these avenues include network compression [40, 107], neural architecture search [174, 26], or parameter quantization [148, 82]. Despite these efforts, few works have explored budgeted training as a solution to accelerate the training of deep neural networks [127, 88, 104, 13]. This setup restricts the number of training iterations available to a given budget.

This thesis addresses budgeted training as an alternative to reduce the computational cost of training CNN while maintaining high performance. Particularly, given a training budget, in Chapter 6 the effectiveness of prioritizing some samples over other is explored, i.e. importance sampling [91, 28, 111], which has not yet been studied under budget restrictions.

Similarly, curriculum learning and self-paced learning organize the samples from easy to difficult to improve model performance by increasing the sample difficulty through training [18, 197, 63, 198]. Particularly, as shown by Weinshall *et al.* [197], curriculum learning improves convergence speed at initial stages of the training because easier samples provide a less noisy estimation of the real gradient. Wu *et al.* [198] experimentally show that curricula are most effective when applied in label noise or budget constrained scenarios. The main drawback, however, is that in most cases the curriculum has to be pre-computed, which is a costly task and often involves manually annotation, transfer learning from a pre-trained model, or pre-training the model in the given dataset. Some approaches reduce this drawback through a simple curriculum [108] or by learning the curriculum during training with a teacher model [86]; these methods do not aim to speed up training but to improve model robustness by weighting the influence of the samples in the training.

Core-set selection approaches, however, are more effective at reducing the computational requirements of training a CNN model [183, 37, 122]. These approaches aim at identifying the subset of samples that will retain most of the performance achieved when training with the full dataset. For that purpose Toneva *et al.* [183] define "forgetting events" as a measure of how forgettable is a sample and show that samples with a higher count of "forgetting events" are more important to the training. Coleman *et al.* [37] demonstrate that proxy models with fewer parameters, despite reaching lower performance still provide useful measures of sample relevance that can be used to train larger models. Mirzasoleiman *et al.* [122] propose a method to select a weighted subset of samples that provably converges to a near-optimal solution. Work in this direction demonstrates that only a portion of the dataset is enough to achieve peak performance. The inconvenience of this approaches is that removing a subset of samples from training brings strong performance degradations. Additionally, the effectiveness in reducing the computation of this approaches is limited by the need for a measure of the contribution of the samples, which requires prior knowledge of the sample contribution to the training, e.g. pre-training the

model.

In the middle ground between curriculum learning and core-set approaches, importance sampling [91, 84, 219] aims at selecting the most important samples at every instant of the training, which has been shown to correspond to the samples with highest gradient magnitude [225, 3]. Different works have proposed several alternative measures to approximate this magnitude: Johnson and Guestrin [87] have shown that the gradients found in the last layer of the model are a good approximation and easier to obtain, Jiang *et al.* [84] use the per-sample loss value during additional forward passes, Chang *et al.* [28] use the probabilities predicted during training for the true class, and Loshchilov and Hutter [111] use the ranking order of these probabilities.

Importance sampling methods propose alternative measures of sample importance that aim at approximating the optimal sampling distribution, which is very computationally demanding to compute. The main problem is that this distribution depends on the state of the model, and measures computed at one iteration become outdated after few updates. Initial attempts to this challenge included several hyperparamters to smooth the estimated distribution [28], additional forward passes to keep the distribution updated [111], or alternative measures to estimate the sampling distribution [7]. Other approaches add complex support techniques to the training to estimate a better distribution: robust optimization [87], repulsive point techniques [217], or a second network trained in parallel with the main model [219]. Recently, several methods leverage a double sampling strategy where a subset of samples is randomly sampled to reduce the search space from the dataset, and then an importance sampling measure is computed in this smaller subset. This technique is often known as random-then-greedy technique [112]. Under this setup Katharopoulos *et al.* [91] define a theoretical bound for the gradient magnitudes which yields a faster computation of the sampling probabilities. With this same sampling strategy, Jiang *et al.* [84] and Ioannou *et al.* [79] use the loss value as a measure of sample importance. Finally, Kawaguchi and Lu [92] use the top-k loss [50]

to perform the backpropagation step using only the samples with highest losses. Note that these methods require a full forward pass every epoch to update the sampling probabilities.

Parallel to these methods, other approaches explore different learning rate schedules to accelerate the training of CNN: a cyclic learning rate that uses higher learning rates at intermediate stages of the training and lower at final stages [164, 163], or a budget-aware schedule that adapts the learning rate value depending on the training stage [104]. Particularly interesting for this thesis is the findings from Li *et al.* [104], that show that in a budgeted scenario, a simple linear decay of the learning rate outperforms the more complex alternatives.

2.2.5 Data augmentation

To increase the variability of the training data, data augmentation techniques introduce transformations to the input images (see Figure 2.8 for an illustration of different augmentation techniques). These often aim to replicate natural variations found in images: translations, rotations, horizontal flips, and color alterations are some of the most frequently used transformations [160]. Data augmentation techniques are being applied to most computer vision applications, but play a key role when training under label and computation constraints. In these scenarios, the overparametrization of neural networks tends to become a liability and hinder generalization by fitting the scarce or corrupted labels. The additional difficulty introduced by data augmentation techniques helps in preventing this happening.

Recently, DeVries *et al.* [43] propose *Cutout*, a data augmentation where contiguous patches of data from the input are dropped, i.e. replaced with zeros. This prevents the network from relying on certain regions of the image over others and results in a spread of the network's attention. Zhang *et al.* [218] propose *mixup*, an interpolation-based data augmentation that uses linear combinations (concretely convex combinations) of inputs and the respective labels for training (see Chapter 4 for a mathematical formulation of *mixup*). This strategy has shown to increase

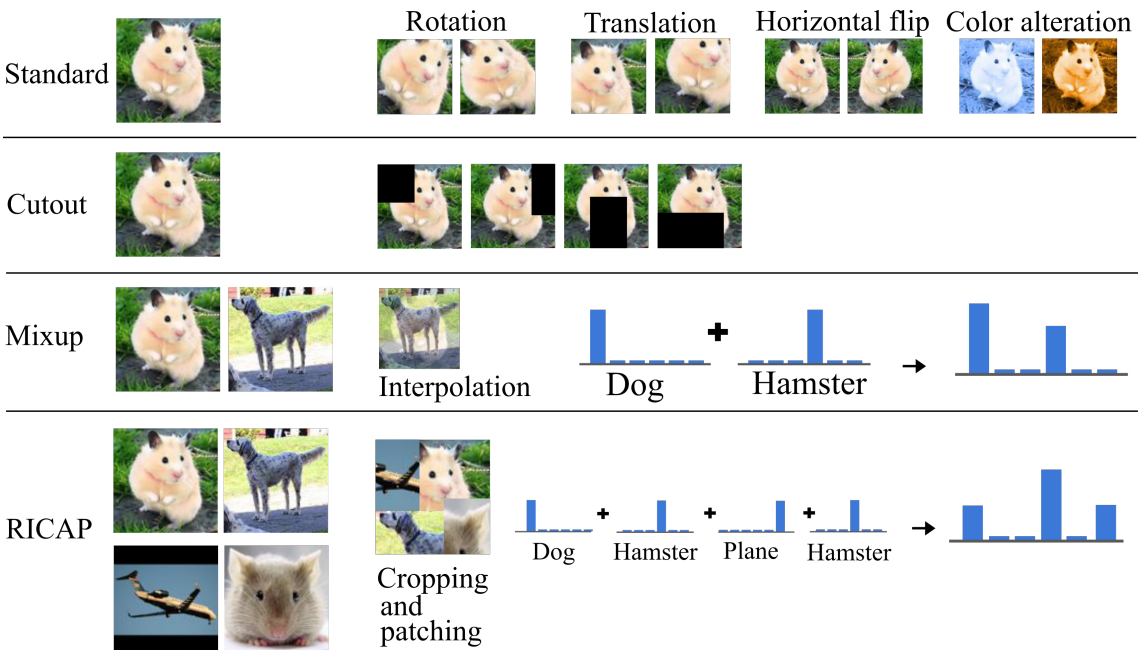


Figure 2.8: Examples of data augmentation techniques. "Standard" refers to affine and color transformations, "Cutout" [43] drops contiguous patches of the image, "Mixup" [218] interpolates two random images (left) and the corresponding labels (right), and "RICAP" [172] crops and pastes patches from different images (left) – typically four – and combines the corresponding labels (right). The combination of labels in "Mixup" and "RICAP" consist of a weighted sum where each label is weighted proportionally to the influence of the corresponding sample in the image.

robustness to label noise and adversarial corruptions, smooth class boundaries, and improve model calibration [179]. Similarly, RICAP [172] combines the advantages of *Cutout* and *mixup* by training on images generated from joining multiple patches and doing the corresponding convex combination of labels. As a more general technique, RandAugment [38] randomly combines commonly used data augmentation techniques as a reduction of the search space of the recently proposed methods that automatically find augmentation policies [75, 39].

2.3 Methodological approach

The experimental chapters of this thesis follow the standard setup found in the computer vision literature. Hence, for each of the chapters, the main components of the training (hyperparameters, network architecture, and datasets) are chosen to enable direct comparison with the most relevant approaches in the literature. In

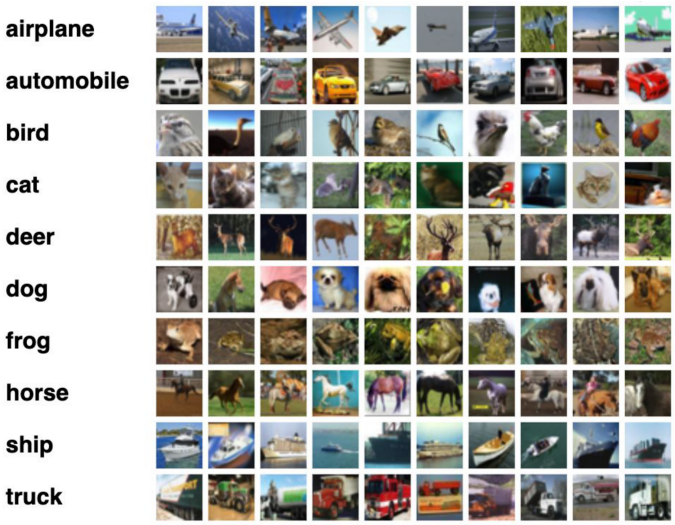


Figure 2.9: Sample images from CIFAR-10. Source: `https://www.cs.toronto.edu/~kriz/cifar.html`

cases where this was not possible, the reported results come from re-running the code provided by the authors of the original papers. The majority of the experiments presented are carried out in some of the toy datasets, CIFAR, STL, and SVHN, to reduce the duration of the experiments, and then the models are further tested in larger and more realistic datasets: Mini-ImageNet, WebVision, and Clothing1M.

2.3.1 Datasets

All the datasets used in this thesis are annotated with per-sample labels indicating the category of the corresponding image. Each of the configurations studied through the thesis requires a different modification of the labels or the images, which are described in Subsection 2.3.2. The following list provides a general overview of the datasets used as provided in the official benchmarks.

- **CIFAR-10 and CIFAR-100** [98]. The CIFAR datasets are composed of 50K RGB images for training and 10K for testing divided into 10 classes in CIFAR-10 and 100 in CIFAR-100. The images in these datasets are collected from ImageNet and down-sampled to 32×32 pixels. These are natural images belonging to classes such as car, dog or bicycle: see Figure 2.9 for example images from each class in CIFAR-10. In the case of CIFAR-100 the classes

Figure 2.10: Sample images from the unlabeled set in STL-10. Source: `https://cs.stanford.edu/~acoates/stl10/`

Figure 2.11: Sample images from SVHN, each row corresponds to a digit category. Source: `http://ufldl.stanford.edu/housenumbers/`

are grouped into 20 meta-classes, 5 classes in each, under a common label, e.g. the meta-class "fish" contains the classes "aquarium fish", "flatfish", "ray", "shark", and "trout".

- **STL-10** [36]. STL-10 was originally designed for unsupervised learning: it contains a training set of 5K RGB images, a testing set of 8K, and an unlabeled set of 100K. The training images are grouped into 10 pre-defined overlapping folds of 1K samples each for a standardized protocol of evaluation that allows reporting the average and standard deviation of the accuracy in the testing set. The images are obtained from ImageNet, down-sampled to 96×96 pixels, and in the training and testing sets, they are divided into 10 categories (similar to those in CIFAR-10). Note that the images in the unlabeled subset belong to a similar distribution of classes (see Figure 2.10) but include classes that are different from the ones present in the training and testing sets.

Figure 2.12: Images selected from the training set of ImageNet.

- **SVHN** [129]. SVHN (Street View House Numbers) is composed of 32×32 RGB images of house numbers obtained from Google Street View images: 73 257 for training and 26 032 for testing distributed into 10 categories corresponding to the digits from 1 to 10 (see Figure 2.11 for sample images of each class). Additionally, there are 531 131 images to use as extra training data. All the samples have class annotations for the central number in the image (often there are other numbers to the sides) and bounding boxes surrounding each of the numbers.

- **Mini-ImageNet** [190]. Created as a computationally affordable benchmark for few-shot learning, Mini-ImageNet, contains 60K RGB images from ImageNet down-sampled to 84×84 pixels distributed into 100 classes (see Figure 2.12). Outside of few-shot learning, it is often split into 50K and 10k for training and testing respectively. Recently a version of this dataset has been released [85] with different types of controlled corruptions to explore label noise algorithms. This is described in detail in Section 2.3.2.

Figure 2.13: Images selected from the training set of WebVision.

- **Tiny-ImageNet**. Similarly to Mini-ImageNet, Tiny-ImageNet is a reduced version of the ImageNet dataset (only used in the experiments of Chapter 5). It contains 100K RGB images for training, 10K for validation, and 10K for testing all down-sampled to 64×64 pixels and distributed across 200 classes.

- **ImageNet (and ImageNet-32/64)** [42]. This dataset corresponds to the data released for the ImageNet competition from 2012 and consist of 1.2M RGB natural images for training and 50K for validation distributed across 1 000 categories (the testing set is historically left out). The images are of different resolutions but, for training and validation, they are often downsized and center-cropped to 256×256 pixels. In the case of ImageNet-32 or ImageNet-64 the images are further down-sampled to allow for faster experimentation. For examples of this and the other ImageNet-based datasets, see Figure 2.12. In this thesis, the down-sampled versions of ImageNet are used in Chapter 5 to explore different label corruption distributions.

- **WebVision** [105]. WebVision contains 2.4M images crawled from the Internet using the 1 000 concepts in ImageNet as query searches (see Figure 2.13). The labels associated to the images come from captions, user tags, or descriptions from the web. This results in a noisy labeled dataset. For experimentation purposes, the 50 first classes are often selected for training, resulting in a set of 137K samples when considering all the data in WebVision, or 63K samples

Figure 2.14: Sample images from Clothing 1M.

when considering only the subset collected by Google. The validation set of ImageNet is often used for testing.

- **Clothing 1M** [204]. Clothing1M contains 1M images for training with potentially noisy labels, 14K for validation, and 10K for testing. Figure 2.14 shows several examples of the images in the dataset. The associated labels are all distributed across 14 classes. Additionally, it contains a subset of 47K correctly annotated samples.

2.3.2 Experimental setup

Each of the chapters of this thesis explores a setup with different restriction in the training conditions: label availability, label corruption, or iteration budget. As a result, the evaluation procedures, data augmentation strategies, and data preparation differ from chapter to chapter. Certain aspects, however, remain constant across setups. Concretely, all approaches use the same optimization method: SGD with momentum of 0.9 and weight decay of 10^{-5}. The training always starts with a random shuffle of the samples, then at each iteration a random batch of samples is selected for training until all the samples have been used. This is repeated for several epochs: one epoch consists of the number of iterations required to see all the samples in the dataset except for budget training in Chapter 6, where each epoch is restricted

to a certain number of iterations. The number of epochs to reach convergence to a reasonable performance depends on the dataset size, training methodology, and supervision availability. This is further specified in the corresponding chapters.

In cases where there is no label corruption (unsupervised learning, semi-supervised learning, and budgeted training), a small subset of labeled data is left out from the training for model selection and hyperparameter tuning. When comparing to other approaches from the literature, this subset is included in the training set and the evaluation is done in a set originally designated for testing. In the case of unsupervised learning with STL-10, this is done using one of the pre-defined folds of the training set. In the presence of label noise, however, hyperparameter tuning cannot be done in a clean validation set because the scenarios studied in this thesis assume that there is no additional clean set of data available. In the case of unsupervised training, the evaluation is done as suggested by the STL-10 authors: after representation learning, the classifier is trained individually in each of the 10 pre-defined folds of the training set and the average performance (and standard deviation) in the test set is reported.

The data preparation in semi-supervised learning is straightforward: the labels of a class-balanced set of samples are discarded to build the unlabeled set. The size of this subset spans across different values to explore the robustness of the methods when different levels of labeled data are available.

In the label noise scenario, however, the data pre-processing depends on the noise distribution and dataset. In each of the types of noise, in-distribution and out-of-distribution, there are two options to introduce corruptions to the datasets: uniformly swapping the labels across classes or following a certain class-dependent distribution. Often, benchmarks like CIFAR-10 and CIFAR-100 are studied as a proxy for more realistic label noise distributions. In these cases, the experiments are limited to in-distribution noise. For uniform noise, the corruptions are randomly introduced, and for non-uniform, the corruptions are guided by the classes in CIFAR-10 ("truck" $\rightarrow$ "automobile", "bird" $\rightarrow$ "airplane", "deer" $\rightarrow$ "horse", "cat" $\rightarrow$ "dog") and by

the meta-classes in CIFAR-100 (label flips are done to the next class circularly within the meta-class) [213]. Recently, Jiang *et al.* [85] and Ortego *et al.* [136] proposed two benchmarks to study realistic noise distributions under a controlled setup: the noisy samples are identified and different levels of noise are available for training. Jiang *et al.* [85] proposed mini-ImageNet-cwln, as an adaptation of Mini-ImageNet where the labels of the samples are used as queries to collect data from the Internet and extend the dataset. The labels associated to the collected samples are potentially corrupted and account for the controlled noise in the dataset. The authors pre-process the data and release it with several levels of label noise, where out-of-distribution samples substitute the clean samples according to the level of noise in each case. Similarly, Ortego *et al.* [136] proposed to use a down-sampled version of ImageNet and split the classes into two distributions and select samples from each of them to adjust the level of in-distribution and out-of-distribution noise for each case.

Finally, for budgeted training in Chapter 6, the datasets are used with all the original labels but the duration of an epoch is redefined to a percentage of the number of iterations required to see all the samples.

2.4 Summary

This chapter contains an introduction to the basic principles of neural networks and their application to computer vision tasks, particularly for image classification under label and budget constraints. The literature review provided in Section 2.2 addresses the most relevant works in the areas of self-supervised and semi-supervised learning, training under label noise, and budgeted training. Finally, in Section 2.3 we introduced the methodology that is followed through the thesis, including the datasets and the data preparation used in the different setups. Int the following chapters, we report and analyze the results obtained throughout the development of this thesis. The corresponding experiments evaluate the validity of the hypotheses proposed in Chapter 1 through the exploration of the corresponding research questions. Additionally, each chapter contains a detailed description of the implementation of

the experiments.

Chapter 3

Unsupervised learning

In this chapter, we explore the self-supervised methodology proposed by Dosovitskiy *et al.* [47], Exemplar-CNN, and introduce a hierarchical agglomerative clustering algorithm in the pipeline to remedy one of its main drawbacks: scalability. As unlabeled data increases, so does the redundancy in the training data, which leads to contradictory SGD updates during training.

While other approaches to self-supervised learning train by solving specific proxy tasks related to particular visual properties of the samples (e.g. orientation or color), Exemplar-CNN aims at learning invariant representations to a large variety of random transformations of the images. This is achieved by training on artificially created surrogate classes that each contain multiple perturbed patches of the original image (see Figure 3.1), which aim at simulating multiple views of a single example; the creation of this surrogate dataset is further explained in Subsection 3.2.2. A convolutional neural network then trains to classify each patch as belonging to the original image. This training method, however, results in contradictory signals for the SGD algorithm that slows down and hiders convergence. By creating one surrogate class per image, Exemplar-CNN trains to classify images from the same true class as belonging to different surrogate classes. This is further aggravated by the redundancy in the dataset: very similar images (near-duplicates) end up belonging to different surrogate classes and providing different update directions for the model weights. This results in slower convergence and performance degradation, particularly when

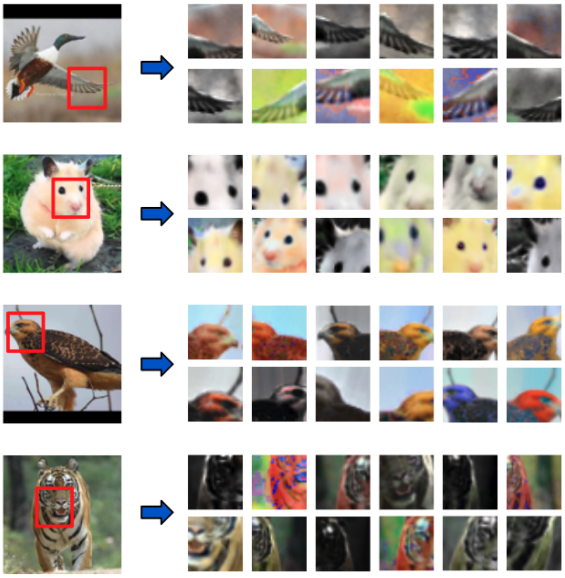


Figure 3.1: Examples of the random transformations in T_i applied to patches extracted from the unlabeled set of STL-10. The location of the extracted patch (red square) is given by the anchor pixel location. The patch is extracted according to a set of different transformations: translations, scaling, rotations, and alterations in color and illumination.

the number of unlabeled samples increases.

Another drawback for the scalability of this method relates to the training architecture: since the classifier has a dimension for each sample in the dataset, as the number of samples increases so does the dimension of the classifier. This results in a weak training signal for the SGD due to the large *softmax* normalization required, and large amounts of parameters in the last layers, that tend to overfit the training data and require careful hyperparameter tuning.

The last drawback of Exemplar-CNN is the strong data augmentation required. While this is essential for this methodology, extreme augmentations become unrealistic and encourage invariance to features that are unlikely in real images. Including such strong augmentations play a strong role in regularizing the model, which further slows down the training. Despite being essential for the algorithm, it is unclear if more realistic alternatives to data augmentation or transformation independent self-supervised approaches would be more effective. This is further discussed as possible future work in Subsection 3.3.1.

This chapter introduces hierarchical agglomerative clustering to the Exemplar-CNN pipeline to compensate for the redundancy in the dataset and to reduce the contradictory updates from similar images placed in different surrogate classes. As a result, the number of unlabeled samples to train Exemplar-CNN can be increased up to 4 times the original size, avoiding degradation of the learned representation while still boosting performance.

While Dosovitskiy *et al.* [47] proposed a clustering algorithm to reduce the collisions between classes, the approach discussed in this chapter consists of a conservative complete linkage (maximal distance) strategy that avoids the clustering of images from different classes and focuses on reducing the redundancy in the dataset by clustering near duplicates. Particularly, while the clustering in Exemplar-CNN [47] generates 6 510 clusters with 10 images per cluster to decrease the number of training surrogate classes, we keep the number of clusters constant and equal to the number of surrogate classes (16 000), each containing approximately 4 images, thus focusing on reducing the possible increase of redundancy in the dataset when new unlabeled data is introduced. This allows Exemplar-CNN to make use of more of the available unlabeled data to learn better representations.

The experiments and results described in this chapter have been published in the Irish Machine Vision and Image Processing Conference (IMVIP), 2019, and the code to replicate the experiments is available at `https://git.io/fjP5u`. The following sections are an adaptation of this work [9] that include further exploration and discussion of Exemplar-CNN. In Section 3.1, the Exemplar-CNN pipeline is introduced, concrete details of the agglomerative clustering are proposed, and the technical details of the combination of both are given. Section 3.2 addresses the experimental part of this part of the thesis: the evaluation methodology and insights on STL-10, the random transformations applied to the images for Exemplar-CNN training, the training setup including the hyperparameters used to train the model and the clustering algorithm, and the experimental results of this chapter. Finally, Section 3.3 provides a discussion on the connection of Exemplar-CNN with more

recent self-supervised alternatives and proposes some directions for future work.

3.1 Method

3.1.1 Exemplar-CNN

We define an initial set of images $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, where n is the number of unlabeled samples $\mathbf{x}_i$ randomly selected to train the Examplar-CNN; concretely in STL-10 $X \subseteq D$ and D is the unlabeled set of 100 000 samples. We then define $T_i = \{T_i^1, \ldots, T_i^k\}$ as the set of transformations applied to the sample $\mathbf{x}_i$: these are compositions of transformations randomly sampled following the descriptions in 3.2.2. By transforming the samples in X we create the surrogate dataset $S = \{S_{\mathbf{x}_1}, \ldots, S_{\mathbf{x}_n}\}$, where $S_{\mathbf{x}_i}$ denotes the set of images resulting of applying T_i to $\mathbf{x}_i$, i.e. $S_{\mathbf{x}_i} = \{T_i^1\mathbf{x}_i, \ldots, T_i^k\mathbf{x}_i\}$: the surrogate class corresponding to $\mathbf{x}_i$.

Consequently, the cross-entropy loss introduced in Subsection 2.1.2 is adapted as

$$\mathcal{L}(W) = -\frac{1}{n \times k} \sum_{i=1}^{n} \sum_{j=1}^{k} \mathbf{y}_i^T \log(h_W(T_i^j \mathbf{x}_i)), \tag{3.1}$$

where $\mathbf{y}_i \in \{0, 1\}^n$ is an n-dimensional one-hot encoded vector that indicates the index of each sample $\mathbf{x}_i$ in X with a 1 (all except the i^{th} element are zero), and $h_W(.)$ corresponds to a convolutional neural network with a *softmax* normalization over the n surrogate classes. See Dosovitskiy *et al.* [47] for an analysis of the convergence guarantees of this loss function.

3.1.2 Agglomerative clustering

Hierarchical agglomerative clustering (HAC) iteratively searches and groups the closest datapoints in a dataset. To decide which observations should be combined, HAC uses a measure of similarity between points or clusters. In the experiments

presented in this thesis we use the Euclidean distance

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_i (\mathbf{p}_i - \mathbf{q}_i)^2}, \tag{3.2}$$

where $\mathbf{p}$ and $\mathbf{q}$ correspond to datapoints in the dataset. When considering distances between two clusters (or a cluster and a datapoint) we use the maximal distance (complete linkage HAC) between all the elements in both clusters:

$$d_{\max}(C_i, C_j) = \max_{\mathbf{p} \in C_i, \mathbf{q} \in C_j} d(\mathbf{p}, \mathbf{q}). \tag{3.3}$$

In this case, C_i and C_j correspond to the set of samples in two clusters (when comparing a datapoint to a cluster, one of these sets contains a single element). The motivation behind this decision (instead of minimum or average distance between clusters) is to further encourage conservative combinations of samples: only the clusters where the most different samples are close enough will be merged.

Once all the samples are inside a single group the clustering can be represented as a dendrogram or hierarchical tree where relationships between samples are defined by the distance between them. Then, by defining a maximum distance between samples or clusters we can select a partition of the data: higher distances will result in fewer clusters but more internally diverse ones, and lower distances in a higher number of clusters and more homogeneous elements in each cluster. Figure 3.2 illustrates this with an example where the dendrogram is cut at a maximum distance to group 11 samples into five clusters.

3.1.3 Agglomerative clustering in Exemplar-CNN

Clustering, in this setup, aims at reducing the number of collisions between surrogate classes belonging to the same true class. This reduces the redundancy of the dataset by grouping together the most similar images and creates a different initial point to build the surrogate dataset: $C = \{C_1, \ldots, C_p\}$, where every C_i corresponds to a cluster of b original images $C_i = \{\mathbf{x}_1, \ldots, \mathbf{x}_b\}$ and generates a new surrogate class

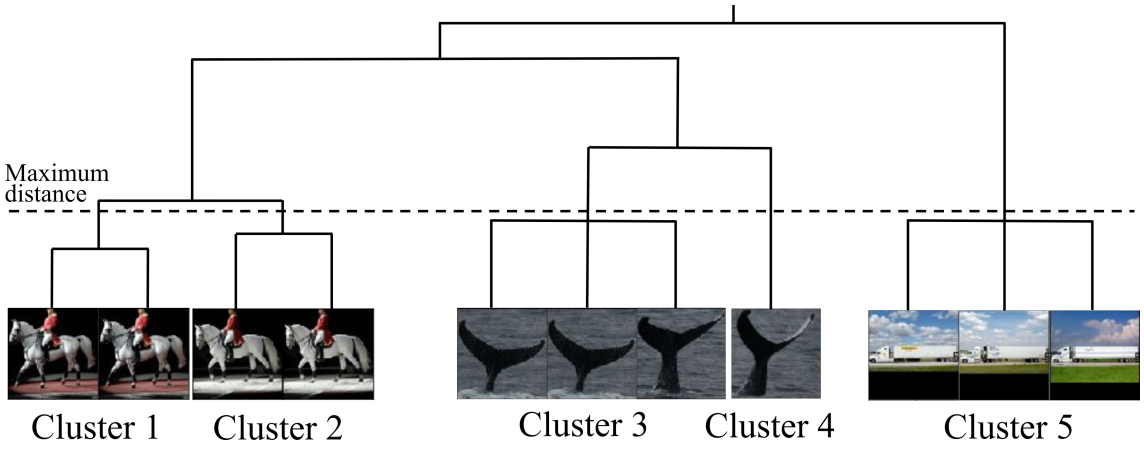


Figure 3.2: Dendrogram representing a hierarchical clustering. The height of the horizontal lines represents the distance between samples of clusters: higher lines link samples that lie far from each other. The dendrogram is cut at a maximum distance between samples (dashed line) to define the clusters.

$\hat{S}_{C_i}$ to train the Exemplar-CNN when applying the transformations T_i:

$$\hat{S}_{C_i} = \{T_1 C_1, \ldots, T_p C_p\}. \tag{3.4}$$

The number of cluster created p and the samples in each cluster b depend on the distance used to cut the dendrogram. This framework allows keeping the number of surrogate classes constant and increasing the amount of unlabeled data without increasing collisions between surrogate classes. Additionally, in the setup proposed in STL-10, this approach introduces robustness to the large number of near-duplicate images in the unlabeled set in STL-10 (see Subsection 3.2.1).

After an initial stage where the CNN trains as proposed in Exemplar-CNN, the model is used as a feature extractor to obtain representations from the unlabeled set and to perform the clustering (the processing of the features is described in Subsection 3.2.1). This results in variable size clusters C_i whose average size depends on the minimum distance allowed inside the clusters. Finally, the Exemplar-CNN approach is applied to the resulting clustered dataset $\hat{S}_{C_i}$.

3.2 Experiments: Clustering for Exemplar CNNs

3.2.1 Dataset: STL-10

Due to its relatively high resolution (96×96), STL-10 allows the exploration of the effect of visual transformations in the Exemplar-CNN framework and still maintains a reasonably low computational cost. Hence, all the experiments in this chapter are carried out on STL-10 (details in Subsection 2.3.1).

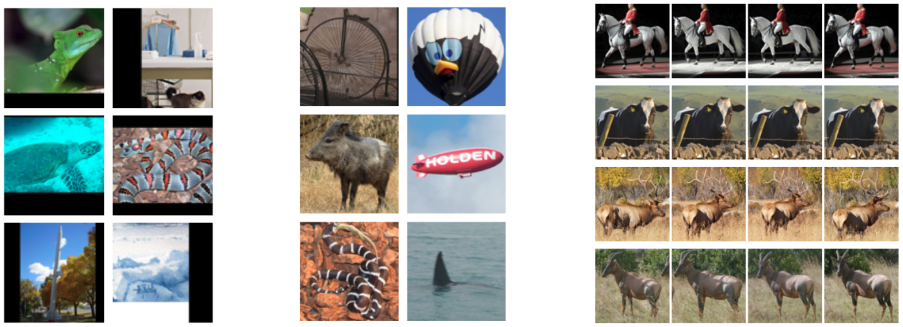
Evaluation protocol

The evaluation protocol follows the guidelines suggested in the STL-10 benchmark [36]: we train a set of linear SVM (one-vs-all) in the features extracted from each of the predefined folds of the training set and report the average accuracy and standard deviation on the testing set. Concretely, the features come from down-sampling each channel of the output features of each convolutional layer to 2×2, flattening them to a single column vector, and concatenating them all together. With the CNN used in this chapter, described in Subsection 3.2.3, this process results in 71 536 dimensional features (Table 3.1 reports the original sizes of the output features of each layer).

Limitations and challenges in STL-10

Visual exploration of the unlabeled set in STL-10 reveals additional challenges that complicate the process of learning self-supervised representations. The following observations should be considered in future research directions:

- **Out-of-distributions samples**: The unlabeled samples are drawn from the ImageNet [42] dataset but from a wider class distribution. Hence, the unlabeled set contains samples that do not belong to any of the 10 classes in the training and tests sets (see Figure 3.3 (b)). This might introduce a shift in the domain statistics between the unlabeled and labeled sets of STL-10 that might increase the difficulty of evaluating the representations learned on the unlabeled set. Additionally, methods that rely on the closed set assumption



(a) (b) (c)

Figure 3.3: Examples of the challenges from STL-10. a) black border artifacts; b) out-of-distribution samples; c) near duplicates.

and try to associate labels to unlabeled samples to further guide the training might be particularly affected by out-of-distribution samples.

- **Artifacts in the images**: Several images have black borders or corners that hinder the quality of the images. This is also the case in the training set and might explain why approaches like *Cutout* [43] are so successful in STL-10: they learn features that are invariant to similar artifacts to those existing in the data. See Figure 3.3 (a) for examples of these artifacts.

- **Near duplicates**: The unlabeled set contains several instances of the same image with imperceptible variations (See Figure 3.3 (c)), which increase the redundancy in the data and exacerbate one of the main limitations of Exemplar-CNNs: class collisions.

3.2.2 Image transformations

The set of transformations $T_i = \{T_i^1, \ldots, T_i^k\}$ generates the surrogate class $S_{\mathbf{x}_i}$ in $S = \{S_{\mathbf{x}_1}, \ldots, S_{\mathbf{x}_n}\}$ by randomly transforming patches around an anchor pixel location in the image $\mathbf{x}_i$. We select the optimal values reported in Dosovitskiy *et al.* [47] for k and n, and generate 100 transformations for each surrogate class for 16 000 images in X. Additionally, for validation purposes, we generate 10 more transformations per surrogate class. The size of the patches correspond to the input size of the CNN and in the main experiments of this chapter, we use 32×32 pixels.

To find relevant anchor locations in the images, we select pixels according to their probability of being an edge in the image: we use the Sobel operator to find edges in the images and assign to each pixel a probability proportional to the edge strength computed for that pixel. The intuition behind this is that strong edges are prevalent near objects and other interesting regions [1]. Figure 3.1 shows some examples of the patches obtained after applying the transformations.

Following Dosovitskiy *et al.* [47] we define each transformation T_i^j as a randomly sampled composition of the following visual alterations:

- **Translations**: the central pixel is translated in the x and y axis a number of pixels given by a Gaussian distribution centered at 0 and with a variance of 0.2 times the size of the original image.

- **Scaling**: the original image is scaled by a factor sampled from a random uniform distribution between 0.7 and 1.4.

- **Rotation and flip**: the original image is flipped horizontally 50% of the time, and rotated by a factor sampled from a random uniform distribution up to 20 degrees.

- **PCA multiplication**: the image pixel values are projected into the 3 dimensional PCA transform computed for the pixels of the image and then, each channel in the PCA space is multiplied, respectively, by a factor between 0.5 and 2 (drawn from a uniform distribution each). Then the image is projected back to the original space.

- **HSV contrast**: the channels of saturation (S) and value (V) for all the pixels in a patch are raised to a power between 0.25 and 4, and the values are multiplied by a factor between 0.7 and 1.4. Then, a value between -0.1 and 0.1 is added to all the pixels from each of the HSV components (using the same value for all pixels in a patch).

[1] We implemented a more semantically driven approach that uses saliency maps (obtained with Pan *et al.* [139]) to select the anchors. We do not report the results with this approach because the improvements are marginal and this introduces a model that required supervision during training.

Table 3.1: Structure of the CNN used in the main experiments. The descriptions and output size dimensions correspond to the experiments with 32×32 input images, for the experiments with 96×96 images, we included two additional convolutional layers before the flatten layer.

Name	Description	Output size
input	32×32 RGB image	$3 \times 32 \times 32$
conv1	92 filters, 5×5, pad = 2, ReLU	$92 \times 32 \times 32$
pool1	Maxpool 3×3 pixels, stride = 2	$92 \times 15 \times 15$
drop1	Dropout, $p = 0.25$	$92 \times 15 \times 15$
conv2	256 filters, 5×5, pad = 2, ReLU	$256 \times 15 \times 15$
pool2	Maxpool 3×3 pixels, stride = 2	$256 \times 7 \times 7$
drop2	Dropout, $p = 0.25$	$256 \times 7 \times 7$
conv3	512 filters, 5×5, pad = 2, ReLU	$512 \times 7 \times 7$
drop3	Dropout, $p = 0.25$	$512 \times 7 \times 7$
flatten	Flatten $512 \times 7 \times 7 \rightarrow 25\,088$	25 088
dense1	Fully connected $25\,088 \rightarrow 1\,024$	1 024
dropD	Dropout, $p = 0.5$	1 024
dense2	Fully connected $1\,024 \rightarrow 16\,000$	16 000
output	Softmax	16 000

3.2.3 Training setup

Exemplar-CNN training

The configuration of the training pipeline for the main experiments follows the descriptions provided by Dosovitsky *et al.* [47] and the corresponding code. In particular, we train a 5-layer CNN composed of 3 convolutional layers and 2 fully connected layers, each followed by a ReLU non-linearity and a dropout layer, with the probability of dropping a weight of $p = 0.25$ for convolutional and $p = 0.5$ for fully connected layers. Table 3.1 provides a detailed description of the architecture. For the experiments with larger images (96×96) from ImageNet, we adapt the CNN and add two more convolutional layers before the fully connected layers to compensate for the increase in the number of parameters in the fully connected layers caused by the input resolution increase.

We train the CNN with SGD with momentum of 0.9 and optimize the model parameters to minimize the loss in Eq. (3.1): the cross-entropy between the *softmax* normalized output of the CNN and the surrogate class label $\mathbf{y}_i$ associated with the

corresponding patch $T_i^j \mathbf{x}_i$. Unlike in a standard supervised configuration, Exemplar-CNN is highly sensitive to the strength of the L_2 regularization and the per-layer value of the learning rate. Given the large number of parameters in the last two layers, a common learning rate for all the layers leads to overfitting through the last fully connected layers and yields poor generalization. Consequently, while the initial learning rate applied to the weights of the convolutional layers and the first fully connected layer is 0.1 (0.2 for the bias), the learning rate applied to the last fully connected layer is 0.025. Similarly, the L_2 regularization strength applied to the weights of the first fully connected layer is 0.004 and 0.016 to the second. The weights and bias of the convolutional layers do not use L_2 regularization. For the experiments with the larger images from ImageNet (96×96), we reduced the weight for the L_2 regularization of the weights of the first and second fully connected layers to 0.003 and to 0.007 respectively. We also applied L_2 regularization to the fifth convolutional layer.

In all the experiments the CNN is trained for 110 epochs and the initial learning rate is reduced as indicated in Dosovitskiy *et al.* [47]: by a factor of 0.4 in the epochs 75 and 92, and by 0.25 in the epochs 85 and 100. The images are pre-processed with a per-channel normalization with the mean and standard deviation from the surrogate dataset generated with the transformations described in Subsection 3.2.2.

Clustering setup

The agglomerative clustering builds a hierarchical tree on the features extracted with the pre-trained Exemplar-CNN from 50% of the data in the unlabeled set in STL-10 (50 000 samples). The processing of the features follow the description in Subsection 3.2.1. We then cut the hierarchical tree to keep clusters with a maximum distance between images of 1.25 – we empirically find this value to provide consistent clusters – and obtain 2 249 clusters of up to 7 samples each. For easier comparison with previous results, we fix the final number of surrogate classes to 16 000: we select 13 751 of the images not clustered after cutting the dendrogram to create each of the

remaining clusters. These are completed with the 3 nearest images from the held-out samples in the unlabeled set (the remaining half of the unlabeled data) according to the cosine distance between the respective features. Finally, we create the new surrogate dataset by applying the transformations described in Subsection 3.2.2 and generating 100 transformations for each image in each cluster (we obtained very similar results when adapting the number of transformations per surrogate class to the elements in each cluster to build a surrogate dataset with a balanced number of samples per class). Note that each cluster has an average of approximately 4 samples.

3.2.4 Results

In this subsection, we present the results of the experimental part of this chapter. The main two baselines are the ExemplarCNN approach proposed by Dosovitskiy *et al.* [47] and our replication of this approach (in the same deep learning programming libraries as the rest of experiments in this thesis). Similarly, the clustering algorithm proposed in this chapter is compared to the clustering approach proposed by Dosovitskiy *et al.* [47] as an evidence of conservative clustering algorithms addressing the main drawback of ExemplarCNN: collisions from near-duplicates samples when this generate different surrogate classes.

Clustering results

Table 3.2 reports the average accuracy and standard deviation of different configurations of the Exemplar-CNN pipeline on the test set of STL-10. *Exemplar-CNN* corresponds to our implementation of Exemplar-CNN [47] and *Clustering* to the proposed clustering algorithm described in this chapter. The increase in accuracy shows that the clustering algorithm allows for an increase in the number of unlabeled samples while keeping the same number of surrogate classes and avoiding the negative effect of increasing the redundancy in the dataset. While the original surrogate dataset used in Dosovitskiy *et al.* [47] contained 1 600 000 image patches, the dataset

Table 3.2: Performance of the clustering algorithm proposed for Exemplar-CNN trained on 16 000 samples from the unlabeled set of STL-10 evaluated on the testing set. The accuracy and standard deviation reported correspond to the performance over the 10 predefined splits of the training set.

Methodology	**Accuracy (%)**
ExemplarCNN [47]	74.2 ± 0.4
Cluster ExemplarCNN [47]	75.4 ± 0.3
ExemplarCNN (ours)	74.14 ± 0.39
Clustering (ours)	**76.42± 0.35**

Table 3.3: Comparative with the state-of-the-art. The methods marked with * use the full training set from STL-10 and evaluate on the testing set. Otherwise, the accuracy and standard deviation reported correspond to the performance over the 10 predefined splits of the training set.

Methodology	**Accuracy (%)**
SWWAE [224]	74.3
ConvClustering [49]	74.10
ExemplarCNN [47]	74.2 ± 0.4
ExemplarCNN (ours)	74.14 ± 0.39
Clustering (ours)	**76.42 ± 0.35**
Cutout* [43]	87.26
IIC* [83]	88.8

resulting from our clustering algorithm is 4 times larger and consist on approximately 6 400 000 patches: 16 000 surrogate classes with an average of 4 images per surrogate class, each transformed 100 times. Note that *Cluster ExemplarCNN* reports the results of the clustering proposed in Dosovitskiy *et al.* [47] that includes 6 500 000 patches distributed in 6 500 clusters of 10 images each.

Comparison to the state-of-the-art

Since the publication of [47], self-supervised methods have been moving towards CIFAR, Mini-Imagenet, and ImageNet, as they contain only images from inside the distribution of expected classes. However, STL-10 is often used as a benchmark for data efficiency where all the images in the training set are used to learn representations that are evaluated in the testing set (*Cutout* and *IIC* in Table 3.3). Conversely, in this thesis we focus on learning representations from unlabeled data and use them as

a prior to train in a very small amount of labeled data: only one of the 10 predefined folds from STL-10 is used for training after the Exemplar-CNN representation learning stage. The top part of Table 3.3 compares the proposed clustering algorithm with relevant state-of-the-art methods for STL-10 and shows that our clustering approach reports the highest performance among the methods that follow this evaluation strategy.

The results reported in the last two rows of Table 3.3 show the importance of labeled data. By using 5 000 labels instead of 1 000 (and no unlabeled data), these methods report accuracies that are up to 12% above the best performance of the approaches that use only 1 000 labeled samples. These results motivate some of the observations in Section 3.3 regarding the advantage that a set of labeled data brings to the training, i.e. semi-supervised learning.

Further exploration of Exemplar-CNN

Table 3.4 provides the results of additional exploratory experiments of the Exemplar-CNN setup. *ImageNet* correspond to the performance of features trained on 96×96 patches obtained from ImageNet, which allowed us to explore a wider range of scaling values in the transformations. Despite the higher resolution of the images, the performance of the features extracted with this model falls far from the Exemplar-CNN trained on STL-10 with 32×32 patches. We hypothesize that this is due to the change in dataset: while the unlabeled set in STL-10 comes from ImageNet, the pre-processing or the selection of the samples might have introduced a domain shift[2] between these two datasets.

As an initial approach to semi-supervised learning, we explore a setup where we consider a labeled pre-defined fold of the training test in STL-10 as a labeled set (1 000 images) and all the unlabeled data as the unlabeled set (100 000). We then train an SVM-based classifier on the features extracted with the pre-trained

[2]Domain shift often refer to the shift of statistics between two sets of samples from two different domains. In this case the sets are from the same domain but obtained through different processes, which might have introduced a shift in the statistics.

Table 3.4: Exploratory experiments on the Exemplar-CNN setup.

Methodology	**Accuracy (%)**
ExemplarCNN (ours)	74.14 ± 0.39
ImageNet	70.7 ± 0.62
Semi-supervised	74.05

Exemplar-CNN on the labeled set and use it to label the unlabeled set. The result reported in Table 3.4 *Semi-supervised* corresponds to a ResNet-50 trained on the resulting dataset: 1 000 labeled samples and 100 000 potentially noisy samples. This experiment shows that Exemplar-CNN could be used as a pretext task for semi-supervised learning or label noise approaches; which are further explored in successive chapters.

Finally, by introducing new transformations, we try to generate new variations that were not considered in the previously described set of transformations. First, we applied elastic transformations [161] aiming to artificially simulate perspective and slight changes in the shape of the objects. Then, we introduced *Cutout* [43] as a data augmentation to encourage Exemplar-CNN to learn features that are invariant to the artifacts present in the unlabeled images. Both of these approaches resulted in a considerable drop in the generalization of the model and reduced its accuracy to 70.19 ± 0.50 in the former and 66.98 ± 0.54 in the later. We also observed a noticeable decrease in the training accuracy of the CNN that suggested that, under these conditions, the network does not have enough capacity to learn meaningful representations.

3.3 Conclusions and discussion

In this section, we introduce advances of self-supervised representation learning in the light of Exemplar-CNN and the framework of learning self-supervised representations by comparing multiple views of unlabeled images. We also discuss the importance of labeled data for pick performance and the dependence of the self-supervised training with the downstream task. Additionally, we introduce some possible avenues for

future work that are further discussed in Chapter 7.

3.3.1 Contrastive learning for the self-supervised multi-view training

Recent approaches for self-supervised representation learning, leverage the underlying idea in Exemplar-CNN of learning features that discriminate each sample from the others. These, however, solve the main challenges of the training through metric learning, concretely through contrastive learning: they substitute the final *softmax* normalization by a distance measure between representations of different views of a sample (see Chapter 2 for more details on the literature regarding contrastive learning approaches).

This paradigm shift solved the inefficient training of Exemplar-CNN: large number of iterations (e.g. disregarding improvements in performance SimCLR [29] in CIFAR-10 needs 4 times fewer iterations than Exemplar-CNN), restrictions in the amount of unlabeled data to be used (to include all the data, Exemplar-CNN needs additional processing such as clustering algorithms), and the *uncommon* configuration of the model (different values for the L_2 regularization or different learning rates per-layer). Contrastive learning solves these challenges while keeping the core idea of learning representations that are invariant to multiple (*all-purpose*) image transformations.

Despite addressing the main challenges in Exemplar-CNN and the improvements in performance and transferability of the representations learned, contrastive learning approaches still present several challenges. The main one is the computational requirements to achieve top performance: most of the methods require a large number of images per mini-batch to better exploit the relationships between image views. SimSiam [32] proposes a promising solution to this challenge and manages to reduce the mini-batch size by contrasting only views of the same sample (removing the negatives in SimCLR) and avoiding degenerated solutions through a stop gradient operation. Similarly, SwAV [27] approaches the problem through an online clustering assignment that resembles to the *softmax* assignments from Exemplar-CNN but

enforces consistency of the representations across assignments. This approach, however, requires a small memory bank when reducing the mini-batch size.

A secondary drawback of the current approach to self-supervised learning is the dependency on data augmentation to generate different views from an image. As shown in InfoMin [181], the optimal views depend on the downstream task (often unknown), which further limits the generalization of these methods to other domains (medical imaging, audio, or text). Additionally, these strong augmentations might slow down the training process, which requires a large number of iterations [181, 29, 59], by further increasing the redundancy in the dataset without leveraging relationships between samples that share common features – in this direction, the cluster assignments in SwAV reduce the number of training iterations considerably.

Contrastive learning approaches to self-supervised learning have been successfully introduced in other tasks such as label noise [137], semi-supervised learning [30], image-to-image translation [140], graph neural networks [69], and video representation learning [67, 2]. This success raises the question: which is the optimal way of populating the feature space for learning better representations? Contrastive learning approaches normalize the representations and spread them over the hypersphere of unity radius by training to push away views from different samples. Despite not having any encouragement to form groups of similar samples, it has been shown [193, 89] that the training paradigm maintains features from similar samples closer while spreading them uniformly through the hypersphere. Intuitively, the end task will dictate the optimal distribution of the features in the space: when aiming at classification we want features that preserve class information and disregard the uniqueness of each sample, but when performing tasks that need some level of reconstruction of the original image, we want features that preserve unique information of each sample rather than commonalities across similar samples. We leave the exploration of the relationship between the distribution of the features and the downstream task for future work.

3.3.2 The need for labels in the target domain

Representation learning without labels has the potential of learning very generic features useful for a large variety of tasks and visual domains. However, once the representations are learned there are very few downstream tasks that directly benefit from them (e.g. retrieval). Most of the target tasks need a specific label projection layer that maps the learned representations to the label space. In that scenario, a small subset of labeled data is still needed.

The need of adapting the representations to the target domain points in the same direction as observations regarding optimal views for contrastive self-supervised learning, and their relationship with the end task. The potential of representation learning could be further exploited in the presence of a subset of labeled data, i.e. semi-supervised learning, to adapt the representations to a particular task. Despite certain self-supervised works evaluating their representations in semi-supervised setups [29, 59, 27], these are not tailored to the specific task; instead, these methods learn generic representations and adapt them through an additional projection head learned on the downstream task. This task-specific bias in the training could also be induced through corrupted labels, i.e. label noise. Even if the guidance is not completely trustworthy it might be sufficient to steer the representations towards the task at hand. These two setups are explored in the Chapter 4 and Chapter 5 and the combination of these with contrastive learning approaches is discussed in Chapter 7 as possible future work.

3.4 Summary

This chapter presented the work done on self-supervised learning. Particularly, it explores agglomerative clustering as a solution to the main drawbacks of a multi-view-based approach to self-supervised learning: Exemplar-CNN. Additionally, this chapters contains insights on the training of Exemplar-CNN, observations on a widely used benchmark for self-supervised learning (STL-10), and a discussion on how recent

approaches to self-supervised learning remedy the drawbacks of Exemplar-CNN and other previous methods.

Chapter 4

Semi-supervised learning

Semi-supervised learning plays a key role in relaxing the human supervision required to train neural networks: by jointly learning from labeled and unlabeled samples, semi-supervised learning leverages vasts amounts of unlabeled data to learn better representations without additional labeling costs. While recent approaches focus on consistency regularization and encourage network predictions to be consistent across different perturbations of unlabeled samples, this chapter explores pseudo-labeling, where network predictions are used as pseudo-labels for the unlabeled samples. Particularly, experiments show that previous approaches at pseudo-labeling overfit incorrect pseudo-labels due to the so-called confirmation bias and fail to learn discriminative representations. This chapter demonstrates that with proper regularization, pseudo-labeling overcomes this challenge and achieves state-of-the-art results in CIFAR-10, CIFAR-100, SVHN, and Mini-ImageNet despite being much simpler than other methods. These results falsify the assumption in recent literature that consistency regularization outperforms pseudo-label based methods [134].

Semi-supervised learning is a relevant task for several domains including vision [134], audio [222], time series [54], and text [123]. Consistency regularization approaches to image classification often exploit the labeled samples through the standard categorical cross-entropy loss function (described in Eq. (2.3)). These approaches, then, leverage the unlabeled samples through an additional consistency term that encourages invariance of network predictions across input perturbations [155,

106, 124], epochs [100], or between other sources of variability in the representations. Pseudo-labeling approaches, however, define pseudo-labels from the model predictions and assign them to the unlabeled samples. The model, then trains on the full dataset, containing the labeled and unlabeled samples, with some adaptation of the cross-entropy loss. These methods often obtain the pseudo-labels directly from the predictions of the network [101], from label graph-based label propagation [81], or leveraging the structure of the feature space [159].

Consistency regularization and pseudo-labeling are vulnerable to degenerate solutions where the guidance for unlabeled samples becomes trivial: despite minimizing the loss function it does not contribute to the training. In other words, the model memorizes the incorrect predictions for the unlabeled samples. These phenomenon is known as confirmation bias [176, 106] or noise accumulation [222] and current approaches to semi-supervised learning apply different techniques to mitigate its effect: a warm-up phase using labeled data [176, 81], uncertainty weighting [159, 106], adversarial attacks [124, 146], or graph-consistency [113, 81]. This bias stems from using incorrect predictions on unlabeled data for training in subsequent epochs, thereby increasing confidence in incorrect predictions and producing a model that will tend to resist new changes. Pseudo-labeling, unlike consistency regularization approaches, does not separate the loss function in two terms which impedes assigning different weights to labeled and unlabeled samples. This makes pseudo-labeling approaches particularly vulnerable to confirmation bias: predictions for unlabeled samples dominate the training and become more confident as the model trains accumulating errors across iterations.

In this chapter we explore pseudo-labeling for semi-supervised learning and show that, contrary to previous attempts [81, 134, 159], simple modifications to prevent confirmation bias lead to state-of-the-art results without appealing to consistency regularization strategies. Following the recent literature [176, 124, 81, 19], we focus the study on class-balanced datasets. As an initial approach to pseudo-labeling, we adapt the relabeling approach proposed by Tanaka *et al.* [175] in the context of

label noise and apply it to the unlabeled samples only. This naive pseudo-labeling, however, is limited by confirmation bias as errors propagate across iterations. To deal with this, the proposed approach includes *mixup* augmentation [218] as an effective regularization that helps calibrate deep neural networks [178] and, therefore, alleviates confirmation bias. *mixup* alone, however, does not guarantee robustness against confirmation bias when the number of labeled samples decreases or when other architectures are used (see Subsection 4.2.2). We show that, when properly introduced, dropout regularization [167] and data augmentation mitigates this issue. The purely pseudo-labeling approach proposed in this chapter achieves state-of-the-art results (see Subsection 4.2.3) without requiring multiple networks [176, 146, 106, 188], nor does it require over a thousand epochs of training to achieve peak performance in every dataset [16, 19], nor does it need many (ten) forward passes for each sample [106]. Compared to other pseudo-labeling approaches, the proposed approach is simpler in that it does not require graph construction and diffusion [81] or combination with consistency regularization methods [159], but still achieves state-of-the-art results.

The work reported in this chapter has been published in the International Joint Conference on Neural Networks (IJCNN) 2020 and the code to replicate the experiments is available at `https://git.io/fjQsC`; the following sections are an adaptation of this work [11]. Section 4.1 introduces the pseudo-labeling approach proposed in [11] and explores how the solutions proposed address confirmation bias and lead to competitive results when the number of available labels decreases. Section 4.2 presents the experimental framework and provide comprehensive studies on the robustness of this approach to different architectures, hyperparameter configurations and availability of labeled samples. Finally, Section 4.3 elaborates on the observations in the experiments and provides a discussion on possible avenues for future work.

4.1 Pseudo-labeling for semi-supervised learning

In this chapter we formulate semi-supervised learning as training a model $h_W(\mathbf{x})$ on a set D of n samples split into the unlabeled set $D_u = \{\mathbf{x}_i\}_{i=1}^{n_u}$ and the labeled set $D_l = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{n_l}$, being $\mathbf{y}_i \in \{0,1\}^c$ the one-hot encoding label for c classes corresponding to $\mathbf{x}_i$ and $n = n_l + n_u$. Following descriptions in Chapter 2, h_W is a convolutional neural network and W represents the model parameters (weights and biases). As this chapter focuses on pseudo-labeling, we assume that a pseudo-label $\tilde{\mathbf{y}}$ is available for the n_u unlabeled samples. We can then reformulate semi-supervised learning as training using $\tilde{D} = \{(\mathbf{x}_i, \tilde{\mathbf{y}}_i)\}_{i=1}^{n}$, being $\tilde{\mathbf{y}} = \mathbf{y}$ for the n_l labeled samples. Then, as in the supervised setup, the model parameters W are optimized using the categorical cross-entropy loss function defined as

$$\mathcal{L}^*(W) = -\frac{1}{n}\sum_{i=1}^{n} \tilde{\mathbf{y}}_i^T \log\left(h_W(\mathbf{x}_i)\right), \tag{4.1}$$

where, as defined in Chapter 2, $h_W(\mathbf{x})$ are the *softmax* probabilities produced by the model and $\log(\cdot)$ is applied element-wise.

A key element in the pseudo-labeling approaches is the generation of the pseudo-labels $\tilde{\mathbf{y}}$ for the n_u unlabeled samples. Previous approaches have used hard pseudo-labels (i.e. scalars encoding the class index instead of one-hot encoded vectors) directly using the network output class [101, 159] or the class estimated using label propagation on a nearest-neighbor graph [81]. We adopt the former approach, but use soft pseudo-labels, as we have seen this outperforms hard labels, confirming the observations noted in [175] in the context of relabeling when learning with label noise. In particular, we store the *softmax* predictions $h_W(\mathbf{x}_i)$ of the network and use them to update the soft pseudo-label $\tilde{\mathbf{y}}$ for the n_u unlabeled samples at the end of every epoch. We proceed as described from the second to the last training epoch, while in the first epoch we use the *softmax* predictions for the unlabeled samples from a model trained in a 10 epochs warm-up phase using the labeled data subset D_u.

Contrastive regularization approaches, conversely, minimize the cross-entropy in

Eq. (2.3) as a supervised term $\mathcal{L}_s$ for the labeled samples, and the mean square error (MSE) between the predictions $h_W(\mathbf{x})$ and a reference $\hat{y}$ as an unsupervised term $\mathcal{L}_u = MSE(h_W(\mathbf{x}), \hat{\mathbf{y}})$ for the unlabeled samples. This often results in a loss function with two terms

$$\mathcal{L} = \mathcal{L}_s + w_t \mathcal{L}_u, \tag{4.2}$$

which correspond to the supervised and unsupervised loss terms. The latter is often weighted with a temporal weight w_t that increases the contribution of the unlabeled samples as the training progresses. This term allows for a stable initial training stage and avoids degenerate solutions where the model overfits the errors made in the unlabeled set.

4.1.1 Proposed approach to pseudo-labeling

The approach proposed in this chapter incorporate the two regularization terms used in [175] to improve convergence. The first, is often referred to as the fairness term and encourages the prediction of all the classes in the distribution in every mini-batch [23, 77, 175, 20]. This deals with the difficulty of converging at early training stages when the predictions of the network are mostly incorrect and the CNN tends to predict the same class to minimize the loss. This is discouraged by adding

$$R_A = \sum_{i=1}^{c} p_i \log\left(\frac{p_i}{\overline{h}_i}\right), \tag{4.3}$$

where p_i is the prior probability distribution for class i and $\overline{h}_i$ denotes the mean *softmax* probability of the model for class i across all samples in the dataset. As in [175], we assume a uniform distribution $p_i = 1/c$ for the prior probabilities (R_A stands for *all classes* regularization) and approximate $\overline{h}_i$ using mini-batches. Note that we assume class-balanced datasets (all the classes have the same number of samples) and expect a mini-batch to be also class-balanced.

The second regularization term sharpens the pseudo-labels assigned to the unlabeled samples, i.e. it concentrates the probability distribution of each soft pseudo-label

on a single class. This is a common term used when training with soft labels [58, 150, 175] that avoids the local optima in which the network might get stuck due to a weak guidance from too flat soft labels:

$$R_H = -\frac{1}{n}\sum_{j=1}^{n}\sum_{i=1}^{c} h_W^i(\mathbf{x}_j)\log\left(h_W^i(\mathbf{x}_j)\right), \tag{4.4}$$

where $h_W^i(\mathbf{x}_j)$ denotes the i class value of the *softmax* output $h_W(\mathbf{x}_j)$ and again using mini-batches (i.e. n is replaced by the mini-batch size) to approximate this term. This second regularization is the average per-sample entropy (R_H stands for entropy regularization), a well-known regularization in semi-supervised learning [57].

Finally, the full semi-supervised loss function is:

$$\mathcal{L} = \mathcal{L}^* + \lambda_A R_A + \lambda_H R_H, \tag{4.5}$$

where λ_A and λ_H control the contribution of each regularization term (see Subsection 4.2.2 for a study of these hyperparameters).

Section 4.2.2 shows that this pseudo-labeling approach adapted from [175] is far from the state-of-the-art for semi-supervised learning, but using the mechanisms proposed in this chapter, described in Section 4.1.2, make pseudo-labeling a competitive alternative.

4.1.2 Confirmation bias in pseudo-labeling

Network predictions are, of course, sometimes incorrect. This situation is reinforced when incorrect predictions are used as labels for unlabeled samples, as is the case in pseudo-labeling. Overfitting incorrect pseudo-labels predicted by the network is known as confirmation bias. It is natural to think that reducing the confidence of the network in its predictions might alleviate this problem and improve generalization. Recently, *mixup* data augmentation [218] introduced a strong regularization technique that combines data augmentation with label smoothing, which makes it potentially useful to deal with this bias. *Mixup* trains on convex combinations of sample pairs

($\mathbf{x}_p$ and $\mathbf{x}_q$) and corresponding labels ($\mathbf{y}_p$ and $\mathbf{y}_q$):

$$\mathbf{x} = \delta \mathbf{x}_p + (1 - \delta)\mathbf{x}_q, \tag{4.6}$$

$$\mathbf{y} = \delta \mathbf{y}_p + (1 - \delta)\mathbf{y}_q, \tag{4.7}$$

where $\delta \in \{0, 1\}$ is randomly sampled from a beta distribution $\mathcal{B}e(\alpha, \beta)$, with $\alpha = \beta$ (e.g. $\alpha = 1$ uniformly selects δ). This combination regularizes the network to favor linear behavior in-between training samples, reducing oscillations in regions far from them. Additionally, Eq. (4.7) can be re-interpreted in the loss as $\mathcal{L}^* = \delta\mathcal{L}_p^* + (1-\delta)\mathcal{L}_q^*$, thus re-defining the loss $\mathcal{L}^*$ used in Eq. (4.5) as:

$$\mathcal{L}^* = -\sum_{i=1}^{n} \delta \left[\tilde{\mathbf{y}}_{i,p}^T \log\left(h_W(\mathbf{x}_i)\right)\right] + (1 - \delta) \left[\tilde{\mathbf{y}}_{i,q}^T \log\left(h_W(\mathbf{x}_i)\right)\right]. \tag{4.8}$$

As shown in [178], overconfidence in deep neural networks is a consequence of training on hard labels and it is the label smoothing effect from randomly combining $\mathbf{y}_p$ and $\mathbf{y}_q$ during *mixup* training that reduces prediction confidence and improves model calibration. In the semi-supervised context with pseudo-labeling, using soft-labels and *mixup* reduces overfitting to model predictions, which is especially important for unlabeled samples whose predictions are used as soft-labels. Note that training with *mixup* generates *softmax* outputs $h_W(\mathbf{x})$ for mixed inputs x, thus requiring a second forward pass with the original images to compute unmixed predictions to be used as pseudo-labels.

Mixup data augmentation alone may be insufficient to deal with confirmation bias when few labeled examples are provided. For example, when training with 500 labeled samples in CIFAR-10 and a mini-batch size of 100, on average, just one clean sample per batch is seen, which is especially problematic at early stages of training where little correct guidance is provided. Oversampling the labelled examples by setting a minimum number of labeled samples per mini-batch k (as done in other works [176, 33, 19, 81]) provides a constant reinforcement with correct labels during

training, reducing confirmation bias and helping to produce better pseudo-labels.

The effect of this oversampling can be understood by splitting the total loss (Eq. 4.1) into two terms, the first depending on the labeled examples and the second on the unlabelled:

$$\mathcal{L}^* = n_l \overline{\mathcal{L}}_l + n_u \overline{\mathcal{L}}_u, \tag{4.9}$$

where n_l and n_u are the number of labelled and unlabelled samples, and the $\overline{\mathcal{L}}_l = \frac{1}{n_l} \sum_{i=1}^{n_l} \mathcal{L}_l^{(i)}$ is the average loss for labeled samples and similarly $\overline{\mathcal{L}}_u$ for the unlabeled samples. The first term is a data loss on the labeled samples and the second can be interpreted as a regularization term that encourages the network to fit the pseudo-labels of the unlabeled samples. When few labeled samples are available, $n_l \ll n_u$, the regularization term dominates the loss, i.e. fitting the pseudo-labels is weighted far higher than fitting the labelled samples. This can be overcome either by upweighting the first term or by oversampling labeled samples. We use the latter strategy as it results in more frequent parameter updates to satisfy the first term, rather than larger magnitude updates.

Subsections 4.2.2 and 4.2.2 experimentally show that *mixup*, a minimum number of samples per mini-batch, and other techniques (dropout and data augmentation) reduce confirmation bias and make pseudo-labeling an effective alternative to consistency regularization.

4.2 Experiments

This subsection contains the experimental part of Chapter 4: first, we describe the experimental framework including dataset preparation and training details; second, we study the effect of the proposed approach on reducing the confirmation bias during semi-supervised training; third, we explore the robustness of the proposed approach under different architectures, levels of labeled samples, and hyperparameter configurations. Finally, in Subsection 4.2.3, we provide a comparison of the results to state-of-the-art approaches.

4.2.1 Experimental framework

Datasets

The experiments presented in this chapter are carried out on four of the image classification datasets introduced in Chapter 2, CIFAR-10, CIFAR-100 [98], SVHN [129] and Mini-ImageNet [190], to validate the proposed approach. As indicated in Subsection 2.3.1, the pre-processing of the datasets for semi-supervised learning consists on splitting the training set into two subsets, labeled and unlabeled, and then discarding the labels of the unlabeled set. Following [134], we leave aside 5 000 samples to use as a validation set for the hyperparameter studies on CIFAR-10 and CIFAR-100 in Subsections 4.2.2 and 4.2.2. However, as done in [16], we add the 5 000 samples back to the training set for comparisons in Subsection 4.2.3, where we report test results (model from the best epoch).

Concretely, for CIFAR-10, CIFAR-100, and SVHN we perform experiments with a number of labeled images $n_l = 250$, 500, 1 000, and 4 000 for CIFAR-10, $n_l =$ 4 000 and 10 000 for CIFAR-100, and $n_l = 250$, 500, and 1000 for SVHN. We use the well-known "13-CNN" architecture [16] for CIFAR-10, CIFAR-100, and SVHN. We also experiment with a Wide ResNet-28-2 (WR-28) [134] and a PreAct ResNet-18 (PR-18) [218] in Subsection 4.2.2 to study the generalization to different architectures. Similarly, as in [81], we use Mini-ImageNet to explore the scalability of the approach to higher resolution images. We experiment with a number of labeled images $n_l =$ 4 000 and 10 000. Following [81], we use a ResNet-18 (RN-18) architecture [72].

Training setup

The experiments in this chapter follow the typical configuration for CIFAR-10, CIFAR-100, and SVHN in [100]. The images are normalized with the dataset mean and standard deviation, and augmented with the standard data augmentation [100]: random horizontal flips, 2 pixel translations for CIFAR and SVHN, and 6 pixel translations for Mini-ImageNet. Additionally, we apply color jitter as in [15] in Subsections 4.2.2 and 4.2.3 for higher robustness against confirmation bias. The

model trains using SGD with momentum of 0.9, weight decay of 10^{-4}, and batch size of 100. Training always starts with a high learning rate (0.1 in CIFAR and SVHN, and 0.2 in Mini-ImageNet) and is divided by ten twice during training. For CIFAR and Mini-ImageNet we train 400 epochs, reduce the learning rate in epochs 250 and 350, and use a 10 epoch warm-up with the labeled data. For SVHN, we train 150 epochs (reducing learning rate in epochs 50 and 100) and use a longer warm-up of 150 epochs to start the pseudo-labeling with good predictions and leading to reliable convergence (experiments in CIFAR-10 with longer warm-up provided results in the same error range already reported). Despite the ablation study in Subsection 4.2.2, we do not select the best configuration for λ_A and λ_H and, for simplicity, just set them to 0.8 and 0.4 as done in [175]. When using dropout, it is introduced between consecutive convolutional layers of ResNet blocks in WR-28, PR-18, and RN-18, while for 13-CNN we introduce it as in [100]. Following [16][1], we use weight normalization [156] in all networks.

4.2.2 Robustness of the proposed approach

This subsection provides experimental evidence of the proposed approach robustness and demonstrates that pseudo-labeling becomes a competitive alternative for semi-supervised learning when properly regularized. Figure 4.1 illustrates how the proposed approach learns successful representations in the "two moons" toy data. The left pane in Figure 4.1 shows the limitations of a naive pseudo-labeling adapted from [175] which learns a linear decision boundary and fails to model the structure of the unlabeled data. The middle pane in Figure 4.1 shows that *mixup* helps the model to learn a smother decision boundary that alleviates confirmation bias and better adapts to the structure of the data. Finally, the right pane in Figure 4.1, shows that *mixup* and a minimum number of labeled samples k per mini-batch further improves the semi-supervised decision boundary.

A naive implementation of pseudo-labeling leads to overfitting the network

[1] `https://github.com/benathi/fastswa-semi-sup`

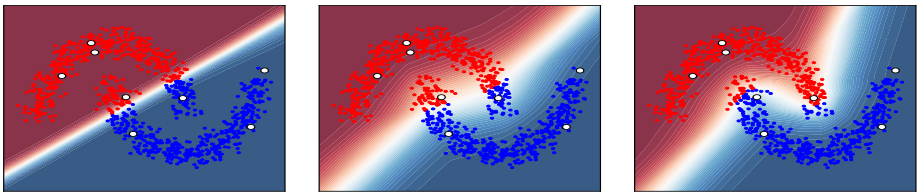


Figure 4.1: Pseudo-labeling in the "two moons" data (4 labels per class) for 1 000 samples. From left to right: no *mixup*, *mixup*, and *mixup* with a minimum number of labeled samples per mini-batch. We use an neural network classifier with one hidden layer with 50 hidden units as in [124]. Best viewed in color.

Table 4.1: Confirmation bias alleviation using *mixup* and a minimum number of k labeled samples per mini-batch. Top: Validation error for naive pseudo-labeling without *mixup* (Cross-entropy), *mixup*, and alternatives with minimum k. Bottom: Study of the effect of k on the validation error.

	CIFAR-10		CIFAR-100
Labeled images	500	4 000	4 000
Cross-entropy	52.44	11.40	48.54
Cross-entropy* ($k = 16$)	35.08	10.90	46.60
Mixup	32.10	7.16	41.80
*Mixup**($k = 16$)	**13.68**	**6.90**	**38.78**

	CIFAR-10		CIFAR-100
Labeled images	500	4 000	4 000
$k = 8$	**13.14**	7.18	42.32
$k = 16$	13.68	**6.90**	**38.78**
$k = 32$	14.58	7.06	39.62
$k = 64$	19.40	8.20	46.28

predictions and high training accuracy in CIFAR-10 and CIFAR-100. Table 4.1 (top) reports *mixup* effect in terms of validation error. Naive pseudo-labeling leads to an error of 11.40 on CIFAR-10 and 48.54 on CIFAR-100 when training with cross-entropy loss for 4 000 labels. This error can be greatly reduced when using *mixup* to 7.16 and 41.80 respectively. However, when further reducing the number of labels to 500 in CIFAR-10, *mixup* is insufficient to ensure low error (32.10). We propose to set a minimum number of samples k per mini-batch to tackle the problem. Table 4.1 (bottom) studies this parameter k when combined with *mixup*, showing that 16 samples per mini-batch works well for both CIFAR-10 and CIFAR-100, dramatically reducing error in all cases (e.g. in CIFAR-10 for 500 labels error is reduced from 32.10 to 13.68).

Confirmation bias causes a dramatic increase in the certainty of incorrect pre-

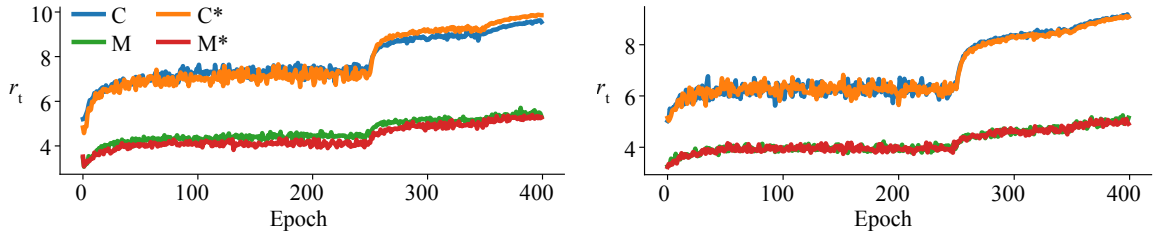


Figure 4.2: Example of certainty of incorrect predictions r_t during training when using 500 (left) and 4 000 (right) labeled images in CIFAR-10. Moving from cross-entropy (C) to *mixup* (M) reduces r_t, whereas adding a minimum number of samples per mini-batch (*) also helps in 500 labels, where M* (with slightly lower r_t than M) is the only configuration that converges, as shown in Table 4.1 (top). Best viewed in color.

dictions during training. To demonstrate this behavior we compute the average cross-entropy of the *softmax* output with a uniform distribution $\mathcal{U}$ across the classes in every epoch t for all incorrectly predicted samples $\{\mathbf{x}_i\}_{i=1}^{m_t}$ as: $r_t = -\frac{1}{m_t}\sum_{i=1}^{m_t}\mathcal{U}^T \log\left(h_W(\mathbf{x}_i)\right)$, where m_t is the number of incorrectly predicted samples. Figure 4.2 shows that *mixup* and minimum k are effective regularizers for reducing r_t, i.e. confirmation bias is reduced. We also experimented with using label noise regularizations [205], but setting a minimum k proved more effective.

Extended hyperparameter study

We explore the effect of the main hyperparameteres α, λ_A, and λ_H in our pseudo-labeling approach. Table 4.2 reports the validation error in CIFAR-10 using 500 and 4 000 labels for, respectively, α and λ_A and λ_H . Note that we keep the same configuration used in Subsection 4.2.2 with $k = 16$, i.e. no dropout or additional data augmentation is used. Table 4.2 results suggest that $\alpha = 4$ and $\alpha = 8$ values might further improve the reported results using $\alpha = 1$. However, we experimented on CIFAR-10 with 500 labels using the final configuration (adding dropout and additional data augmentation) and observed marginal differences (8.54 with $\alpha = 4$, which is within the error range of the 8.80 ± 0.45 obtained with $\alpha = 1$) shown in Table 4.4, thus suggesting that stronger *mixup* regularization might not be additive to dropout and extra data augmentation in our case. Table 4.2 shows that our configuration ($\lambda_A = 0.8$ and $\lambda_H = 0.4$) adopted from [175] is very close to the best

Table 4.2: Validation error for different values of the α parameter from *mixup*, λ_A, and λ_H. Bold indicates lowest error. Underlined values indicate the results of the configuration used.

Labeled images:	500				4 000			
α	0.1	1	4	8	0.1	1	4	8
	23.18	<u>13.68</u>	**10.60**	11.04	8.58	<u>6.90</u>	**6.56**	6.68
λ_A/λ_H	0.1	0.4	0.8	2	0.1	0.4	0.8	2
0.1	22.94	29.64	60.76	83.96	7.22	**6.88**	7.74	33.98
0.4	20.92	**12.88**	17.62	38.40	7.18	6.96	7.18	8.82
0.8	23.50	<u>13.68</u>	14.72	25.92	7.24	<u>6.90</u>	7.18	8.78
2	31.30	14.80	14.62	23.40	8.16	7.28	7.40	8.64

performance in this experiment where marginal improvements are achieved. More careful hyperparameter tuning might slightly improve the results here, but the default configuration is already good and generalizes well across datasets.

Generalization to different architectures

Recent examples in the literature [97] show that moving from one architecture to another changes which methods appear to have a higher potential. In particular, Kolesnikov *et al.* [97] show that the presence of skip-connections in ResNet architectures directly affects the quality of the representations learned. Similarly, Ulyanov *et al.* [185] showed that different architectures lead to different and useful image priors, highlighting the importance of exploring different networks. We therefore explore our method on two more architectures: a Wide ResNet-28-2 (WR-28) [153] and a PreAct ResNet-18 (PR-18) [71]. The former contains around 1.5M parameters and is typically used in semi-supervised learning [134], and the latter 11M parameters and is used in the context of learning representations in the presence of label noise [218].

Table 4.3 presents the results for the AlexNet-type architecture, 13-CNN, and these two ResNet-type architectures, WR-28 and PR-18. The pseudo-labeling proposed in this chapter with *mixup* and $k = 16$ (M*) works well for 4 000 and 500 labels across architectures, except for 500 labels for WR-28 where the error increases considerably (29.50). This is due to a stronger confirmation bias in which labeled samples are not properly learned, while incorrect pseudo-labels are fit. Interestingly,

Table 4.3: Validation error across architectures is stabilized using dropout (p) and data augmentation (A).

Labeled images	500	4 000
13-layer		
M*	13.68	6.90
M* ($p = 0.1$)	12.62	6.58
M* ($p = 0.3$)	11.94	6.66
M* ($p = 0.1, A$)	**9.16**	**6.22**
WR-28		
M*	29.50	**6.40**
M* ($p = 0.1$)	14.14	7.06
M* ($p = 0.3$)	30.56	11.44
M* ($p = 0.1, A$)	**10.94**	6.74
PR-18		
M*	**13.90**	**5.94**
M* ($p = 0.1$)	14.78	5.90
M* ($p = 0.3$)	14.78	6.62
M* ($p = 0.1, A$)	14.96	6.32

PR-18 (11M parameters) is more robust to confirmation bias than WR-28 (1.5M parameters), while the 13-layer network (3M parameters) has fewer parameters than PR-18 and achieves better performance. This suggests that the network architecture plays an important role, being a relevant prior for semi-supervised learning with few labels.

Experimental results show that by introducing dropout and additional data augmentation the proposed pseudo-labeling performs well across architectures. Particularly, Table 4.3 shows that dropout, $p = 0.1$ and $p = 0.3$, helps in achieving better convergence in CIFAR-10 and that additional data augmentation, i.e. color jitter (as described in Subsection 4.2.1), further contributes in reducing the error. Note that to maintain pseudo-label quality, we disable dropout when computing them in the second forward pass. Similarly, we observe that disabling the data augmentation in the second forward pass further improves performance. We use this configuration for the comparison with the state-of-the-art in Subsection 4.2.3.

Table 4.4: Test error in CIFAR-10 and CIFAR-100 for the proposed approach using the 13-CNN network. (*) denotes that we have run the algorithm. Bold indicates lowest error. We report average and standard deviation of 3 runs with different labeled/unlabeled splits.

	CIFAR-10			CIFAR-100	
Labeled images	500	1 000	4 000	4 000	10 000
Supervised (C)*	43.64 ± 1.21	34.83 ± 1.15	19.26 ± 0.26	54.49 ± 0.53	41.14 ± 0.26
Supervised (M)*	37.60 ± 0.65	28.59 ± 1.21	15.94 ± 0.26	52.70 ± 0.28	39.42 ± 0.37
Consistency regularization methods					
Π model [176]	-	-	12.36 ± 0.31	-	39.19 ± 0.36
TE [176]	-	-	12.16 ± 0.24	-	38.65 ± 0.51
MT [176]	27.45 ± 2.64	19.04 ± 0.51	11.41 ± 0.25	45.36 ± 0.49	36.08 ± 0.51
Π model-SN [113]	-	21.23 ± 1.27	11.00 ± 0.13	-	37.97 ± 0.29
MA-DNN [33]	-	-	11.91 ± 0.22	-	34.51 ± 0.61
Deep-Co [146]	-	-	9.03 ± 0.18	-	38.77 ± 0.28
MT-TSSDL [159]	-	18.41 ± 0.92	9.30 ± 0.55	-	-
MT-LP [81]	24.02 ± 2.44	16.93 ± 0.70	10.61 ± 0.28	43.73 ± 0.20	35.92 ± 0.47
MT-CCL [106]	-	16.99 ± 0.71	10.63 ± 0.22	-	34.81 ± 0.52
MT-fast-SWA [16]	-	15.58 ± 0.12	9.05 ± 0.21	-	34.10 ± 0.31
ICT [188]	-	15.48 ± 0.78	7.29 ± 0.02	-	-
Pseudo-labeling methods					
TSSDL [159]	-	21.13 ± 1.17	10.90 ± 0.23	-	-
LP [81]	32.40 ± 1.80	22.02 ± 0.88	12.69 ± 0.29	46.20 ± 0.76	38.43 ± 1.88
Ours*	**8.80 ± 0.45**	**6.85 ± 0.15**	**5.97 ± 0.15**	**37.55 ± 1.09**	**32.15 ± 0.50**

4.2.3 Comparison with the state-of-the-art

This subsection compares our pseudo-labeling approach against related methods that use the 13-CNN architecture [176] in CIFAR-10 and CIFAR-100: Π model [100], *Temporal Ensemble* (TE) [100], *Mean Teacher* (MT) [176], Π model-SN [113], MA-DNN [33], Deep-Co [146], TSSDL [159], LP [81], CCL [106], fast-SWA [16] and ICT [188]. The results reported in Table 4.4 and Table 4.5 are divided into methods based on consistency regularization and pseudo-labeling. The methods that combine pseudo-labeling with consistency regularization, e.g. MT, are included in the consistency regularization set.

The proposed pseudo-labeling approach outperforms the consistency regularization alternatives, as well as the other pseudo-labeling approaches (and their combinations with consistency regularization methods) in CIFAR-10 and CIFAR-100. Similarly, in SVHN, the proposed approach outperforms most state-of-the-art methods, especially when there are very few labels available. These results demonstrate

Table 4.5: Test error in SVHN for the proposed approach using the 13-CNN network. (*) denotes that we have run the algorithm. Bold indicates lowest error. We report average and standard deviation of 3 runs with different labeled/unlabeled splits.

Labeled images	250	500	1 000
Supervised (C)*	43.60±3.35	22.67±2.80	13.32±0.89
Supervised (M)*	53.15±6.54	20.74±0.80	11.66±0.17
Consistency regularization methods			
Π model [176]	9.69 ± 0.92	6.83 ± 0.66	4.95 ± 0.26
TE [176]	-	5.12 ± 0.13	4.42 ± 0.16
MT [176]	4.35 ± 0.50	4.18 ± 0.27	3.95 ± 0.19
Π model-SN [113]	5.07 ± 0.25	4.52 ± 0.30	3.82 ± 0.25
MA-DNN [33]	-	-	4.21 ± 0.12
Deep-Co [146]	-	-	3.61 ± 0.15
MT-TSSDL [159]	4.09 ± 0.42	3.90 ± 0.27	**3.35 ± 0.27**
ICT [188]	4.78 ± 0.68	4.23 ± 0.15	3.89 ± 0.04
Pseudo-labeling methods			
TSSDL [159]	5.02 ± 0.26	4.32 ± 0.30	3.80 ± 0.27
Ours*	**3.66 ± 0.12**	**3.64 ± 0.04**	3.55 ± 0.08

the generalization of the proposed approach compared to other methods that fail when decreasing the number of labels. Furthermore, Table 4.6 demonstrates that the proposed approach successfully scales to higher resolution images, obtaining an over 10 point margin on the best related work in Mini-ImageNet. Note that all supervised baselines are reported using the same data augmentation and dropout as in the proposed pseudo-labeling.

Table 4.7 (right) compares the proposed pseudo-labeling against recent consistency regularization approaches that incorporate *mixup*. Our approach with WR-28 outperforms ICT [188] and is competitive with MM [19] for 500 and 4 000 labeled samples. When training with PR-18, we reach a reasonable convergence for 500 and 4 000 labels, whereas for 250 we do not. Finally, the 13-CNN architecture robustly converges to state-of-the-art results even for 250 labels where we obtain 9.37 test error. Therefore, these results suggest that it is worth exploring the relationship between the number of labels, dataset complexity and architecture type. As shown in Subsection 4.2.2, dropout and additional data augmentation help with 500 labels per class across architectures, but are insufficient for 250 labels. Better data augmentation [74] or self-supervised pre-training [149] might overcome this challenge. However, it is

Table 4.6: Test error in Mini-ImageNet. (*) denotes that we have run the algorithm. Bold indicates lowest error. We report average and standard deviation of 3 runs with different labeled/unlabeled splits.

Labeled images	4 000	10 000
Supervised (C)*	75.69 ± 0.24	63.24 ± 0.33
Supervised (M)*	72.03 ± 0.21	59.96 ± 0.40
Consistency regularization methods		
MT [176]	72.51 ± 0.22	57.55 ± 1.11
MT-LP [81]	72.78 ± 0.15	57.35 ± 1.66
Pseudo-labeling methods		
LP [81]	70.29 ± 0.81	57.58 ± 1.47
Ours*	**56.49 ± 0.51**	**46.08 ± 0.11**

Table 4.7: Test error in CIFAR-10 with few labeled samples. (*) denotes that we have run the algorithm. Bold indicates lowest error. We report average and standard deviation of 3 runs with different labeled/unlabeled splits.

Labeled images	250	500	4 000
MM (WR-28) [19]	**11.08 ± 0.87**	**9.65 ± 0.94**	**6.24 ± 0.06**
ICT* (WR-28) [188]	52.19 ± 1.54	42.33 ± 0.08	7.26 ± 0.04
Ours* (WR-28)	24.81 ± 5.35	14.25 ± 0.86	6.28 ± 0.3
Ours* (13-CNN)	**9.37 ± 0.12**	**8.80 ± 0.45**	5.97 ± 0.15
Ours* (PR-18)	23.86 ± 4.82	12.16 ± 1.06	**5.86 ± 0.17**

already interesting that a straightforward modification of pseudo-labeling, designed to tackle confirmation bias, gives a competitive semi-supervised learning approach, without any consistency regularization, and future work should take this into account.

4.3 Conclusions and discussion

In this chapter, we explore semi-supervised learning in image classification through pseudo-labeling. The experiments demonstrate that when confirmation bias is addressed, pseudo-labeling approaches are a suitable alternative to the dominant approach in the literature, consistency regularization, and achieve state-of-the-art results in several benchmarks. Concretely, our approach uses the network predictions as soft pseudo-labels for unlabeled data and alleviates confirmation bias by introducing *mixup* augmentation, a minimum number of labeled samples per batch, dropout, and data augmentation.

In light of further exploration of pseudo-labeling for semi-supervised learning, we observe that the quality of the pseudo-labels is a key component to stabilize training for an extremely low number of labeled samples: strong data augmentation strategies and dropout for the pseudo-label computation damage convergence by providing imprecise labels. This suggests that pseudo-labels could be further improved with techniques that refine network predictions such as a momentum network to compute pseudo-labels as in MT [176] or ensembles of the model in different epochs as in TE [100].

The proposed approach, however, is simpler and more accurate than most recent alternatives. Yet, we believe that synergies between consistency regularization and pseudo-labeling could result in interesting and efficient approaches to semi-supervised learning, as suggested by recent tendencies to "holistic" approaches that combine several techniques from the literature [19, 20]. Similarly, future lines of work that remain unexplored relate to two widely adopted assumptions in the literature: unlabeled samples belong to the class distribution in the labeled set (i.e. the unlabeled set is free from out-of-distribution samples), and the number of samples is balanced across classes. Exploration in this direction is of utmost importance for enabling further applicability of current methods to realistic semi-supervised scenarios.

Finally, it is worth noting that the performance degradation stemming from incorrect pseudo-labels also appears in supervised applications where labels cannot be fully trusted, i.e. as label noise. Despite presenting a different source of corruption, approaches to label noise could benefit from the insights reported in this chapter. The exploration presented in the following chapter addresses this setup, by proposing a robust method to learn under the presence of label noise, and providing further insights into the training of convolutional neural networks under supervision constraints.

4.4 Summary

This chapter addresses the approach developed for semi-supervised image classification and includes extensive experimental details: descriptions of the technical framework for the experiments (Subsection 4.2.1), studies of the proposed approach on several benchmarks (Subsection 4.2.3), and exploration of the robustness to different hyperparameters, levels of labeled samples, and model architectures (Subsection 4.2.2). In particular, we identify a crucial challenge in pseudo-labeling approaches to semi-supervised learning, confirmation bias, and propose a solution to reduce its effect on representation learning. Experimental results show that when accounting for confirmation bias, pseudo-labeling approaches can surpass state-of-the-art consistency regularization based approaches.

Chapter 5

Label noise

The recent spike in interest in CNN training under label corruptions has resulted in robust methods able to exploit vast amounts of visual data [103, 196, 110] that avoid the exhaustive labeling process of the classical fully supervised learning. By relaxing this dependence on high-quality labels, label noise training reduces the costs in data annotation stages. To this end, research focuses on synthetic scenarios designed to replicate the complex characteristics of the noise in the data, i.e. label corruptions as introduced in Chapter 2 (Subsection 2.3.2). These synthetic scenarios allow for the replication of experiments, exploration different levels of label corruption, and study the behavior of the models for clean and noisy samples individually (since noisy samples are identified beforehand). Two main scenarios dominate the research landscape where label corruptions are introduced following uniform or non-uniform distribution. In the former, known as symmetric noise, labels are randomly swapped between classes. In the latter, asymmetric noise, labels are swapped to another class depending on the similarity between this new class and the true class of the image; this is a class-dependent noise. Section 5.3 further explores the assumption that real-world label noise can be characterized by these two distributions and introduces research that explores alternative approaches [204, 105, 85].

As noted in [216], the high number of parameters in CNNs allows them to memorize datasets with random labels when trained for enough time. This results in high performance on the training set, but poor on the testing set. In a clean

dataset, however, we expect that competitive performance during training precedes competitive performance in testing, i.e. features generalize outside the known data to the testing set. It is useful to think of this problem as generalization vs. memorization, and these two concepts are very intertwined in label noise. Distinctions between these two concepts become evident on symmetric noise distributions, where noisy samples do not share structure between them and noisy labels do not depend on the features in the image. In this scenario, CNNs easily learn features that generalize across clean samples, which is reflected as a decrease in the loss value of the corresponding samples. Later in training, however, to keep reducing the average loss, the CNN starts memorizing the noisy samples, which results in features that do not generalize across samples (much less to the testing set). This is, consequently, reflected in the loss values of the noisy samples, which decrease later in training than that of clean samples (Figure 5.1). Note that this chapter is focused on label noise, but these observations might extend to other sources of corruption in the dataset such as biases [6] or fake content [182].

This difference in the behavior of the loss values of clean and noisy samples is less evident in asymmetric noise distributions. This is probably because incorrect labels in asymmetric noise contain information of the similarity between classes, which allows the CNN to leverage certain features shared by noisy samples. As a consequence, these samples are memorized earlier and make the distinction between clean and noisy samples more challenging when considering only the loss values (this is further discussed in Subsection 5.3.1 and illustrated by Figure 5.6).

Regardless of the noise distribution, CNNs learn corrupted representations when trained under these conditions. Through the development of this thesis we studied in [136] interactions between different noise distributions and the learning process of CNNs, and later in [137] we propose a robust alternative to train CNN in the presence of label noise. The most relevant findings in these publications are discussed in Section 5.3. This chapter, however, explores the research published in the International Conference on Machine Learning 2019, in [10] (source code is available

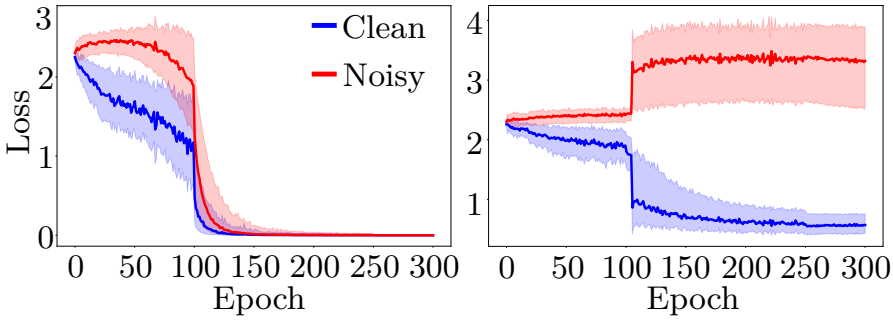


Figure 5.1: Cross-entropy loss on CIFAR-10 under 80% label noise for clean and noisy samples. Left: training with cross-entropy loss results in fitting the noisy labels. Right: using our proposed objective prevents fitting label noise while also learning from the noisy samples. The heavy lines represent the median loss values and the shaded areas are the interquartile ranges.

at `https://git.io/fjsvE`), and extends it by including further exploration of the approach and additional regularization terms that stabilize the training across noisy scenarios. Of course, the chapter is written in the light of the latest observations from [136] and [137] on the effect of label noise in the training of CNNs.

In particular, this chapter presents an unsupervised noise modeling strategy that estimates the probability of a sample being noisy and leverages it to guide a bootstrapping mechanism by weighting the contribution of the label and the prediction. Additionally, we adapt this loss function to the interpolation training strategy from *mixup* [218] to build a final approach that is robust to high levels of noise in the dataset.

5.1 Proposed approach: Dynamic bootstrapping for label noise

The formulation of image classification under label noise follows the notation described in previous chapters: a model $h_W(\mathbf{x})$ corresponds to a CNN with W parameters that trains on a set of training examples $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{n}$ with $\mathbf{y}_i \in \{0,1\}^c$ being the one-hot encoding ground-truth label corresponding to $\mathbf{x}_i$. In particular, for label noise, the label $\mathbf{y}_i$ can be noisy, i.e. refer to a class that does not correspond to the image in $\mathbf{x}_i$. Note that for the out-of-distribution noise mentioned in the introduction

of the chapter (and further discussed in Section 5.2.6), the true label for $\mathbf{x}_i$ might not be in the distribution of labels observed in the training set. The CNN trains on the set $\mathcal{D}$ to minimize the categorical cross-entropy loss function (earlier defined in Eq 2.3):

$$\mathcal{L}(W) = \frac{1}{n}\sum_{i=1}^{n} \ell_i(W) = -\sum_{i=1}^{n} \mathbf{y}_i^T \log\left(h_W(\mathbf{x}_i)\right), \tag{5.1}$$

where $h_W(\mathbf{x})$ are the *softmax* probabilities produced by the model and $\log(\cdot)$ is applied element-wise, i.e. to the probabilities predicted for each sample.

In this section we review the approach to label noise proposed in [10]: the label noise modeling technique, the adaptations added to the loss in (5.1), and the regularizations that help handling label noise. Additionally, in Subsection 5.1.3, we provide insights into the regularizaztion role of *mixup* in label noise. For notational simplicity, we use $\ell_i(W) = \ell_i$ and $h_W(\mathbf{x}_i) = \mathbf{h}_i$ in the remainder of the chapter.

5.1.1 Identifying corrupted labels: noise detection

In this subsection, we explore the behavior of the loss function when training under label noise and how this can be used to separate clean and noisy samples. Other approaches to noise detection that we have explored through the development of the thesis are based on loss values of a relabeling stage [136] that aim at mitigating the memorization of the noisy labels; as well as a discrepancy measure between the labels of a set of neighbors [137]. Despite focusing on the loss value, this chapter provides in Section 5.3 a discussion and insights into these other two techniques for detecting noisy samples.

The main contribution of the approach proposed in [10] lies in the identification of noisy samples in the dataset $\mathcal{D}$ and the proper adaptation of the loss to control the degree to which parameter updates are influenced by the label of a given sample (see Subsections 5.1.2 and 5.1.3). The approach relies on the observation that clean samples are learned earlier in the training, which means that, in those stages, noisy samples have higher loss values on average (Figure 5.1 left). This allows clean and noisy samples to be distinguished from the loss distribution alone. As shown in [216],

CNN trained with SGD are able to fit all samples with random labels. However, the memorization of noisy labels does not happen until substantial progress has been made in fitting the clean ones. Since one can infer from the loss value if a sample is more likely to be clean or noisy, this chapter proposes to use a mixture distribution model for this purpose. These could be used to split the dataset into noisy and clean samples or to obtain the probability of a sample being clean or noisy. The former has shown to be useful in discarding noisy samples or their labels to apply semi-supervised algorithms [136], while the latter is exploited in this chapter to obtain a measure of how reliable is a sample, i.e. the probability of being noisy.

Mixture models are a widely used unsupervised modeling technique [168, 142, 116], with the Gaussian Mixture Model (GMM) [142] being the most popular. The probability density function (pdf) of a mixture model of k components on the loss ℓ is defined as:

$$p(\ell) = \sum_{j=1}^{k} \lambda_j \, p(\ell \mid j) \,, \tag{5.2}$$

where λ_j are the mixing coefficients for the convex combination of each individual pdf $p(\ell \mid j)$. In our case, we can fit a two component GMM (i.e. $k = 2$ and assuming ℓ follows a Gaussian distribution as $\ell \sim \mathcal{N}\left(\mu_j, \sum_j\right)$) to model the distribution of clean and noisy samples (Figure 5.2). Unfortunately, the Gaussian is a poor approximation to the clean set distribution, which exhibits high skew toward zero. The more flexible beta distribution [116] allows modeling both symmetric and skewed distributions over $[0, 1]$; the beta mixture model (BMM) better approximates the loss distribution for mixtures of clean and noisy samples (Figure 5.2). Empirically, we also found the BMM improves ROC-AUC for clean-noisy label classification over the GMM by around 5 points for 80% label noise in CIFAR-10 when using the training objective in Section 5.1.3 (see 5.2.3). The beta distribution over a (max) normalized loss $\ell \in [0, 1]$ is defined to have pdf:

$$p(\ell \mid \alpha, \beta) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\,\Gamma(\beta)} \ell^{\alpha - 1} \left(1 - \ell\right)^{\beta - 1} \,, \tag{5.3}$$

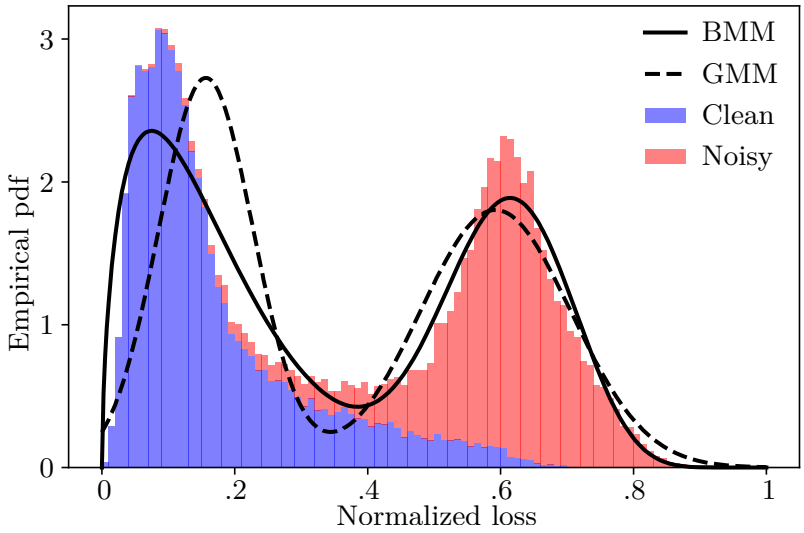


Figure 5.2: Empirical PDF and estimated GMM and BMM models for 50% label noise in CIFAR-10 after 10 epochs with standard cross-entropy loss and learning rate of 0.1 (the remaining hyperparameters are provided in Subsection 5.2.1). Clean and noisy samples are colored for illustrative purposes. The BMM model better fits the skew toward zero loss of the clean samples.

where $\alpha, \beta > 0$ and $\Gamma(\cdot)$ is the Gamma function, and the mixture pdf is given by substituting the above into Eq. (5.2).

We use an Expectation Maximization (EM) procedure to fit the BMM to the observations. Specifically, we introduce latent variables $\gamma_j(\ell) = p(j|\ell)$ which are defined to be the posterior probability of the point ℓ having been generated by mixture component j. In the E-step we fix the parameters $\lambda_j, \alpha_j, \beta_j$ and update the latent variables using Bayes rule:

$$\gamma_j(\ell) = \frac{\lambda_j \, p(\ell \mid \alpha_j, \beta_j)}{\sum_{m=1}^{K} \lambda_m \, p(\ell \mid \alpha_m, \beta_m)}. \tag{5.4}$$

Given fixed $\gamma_j(\ell)$, the M-step estimates the distribution parameters α_j, β_j using a weighted version of the method of moments:

$$\beta_j = \frac{\alpha_j \left(1 - \bar{\ell}_j\right)}{\bar{\ell}_j}, \;\; \alpha_j = \bar{\ell}_j \left(\frac{\bar{\ell}_j \left(1 - \bar{\ell}_j\right)}{s_j^2} - 1 \right) \tag{5.5}$$

with $\bar{\ell}_j$ being a weighted average of the loss values $\{\ell_i\}_{i=1}^n$ corresponding to each

training sample $\{x_i\}_{i=1}^n$, and s_j^2 being a weighted variance estimate:

$$\bar{\ell}_j = \frac{\sum_{i=1}^n \gamma_j(\ell_i)\, \ell_i}{\sum_{i=1}^n \gamma_j(\ell_i)}, \tag{5.6}$$

$$s_j^2 = \frac{\sum_{i=1}^n \gamma_j(\ell_i)\, \left(\ell_i - \bar{\ell}_j\right)^2}{\sum_{i=1}^n \gamma_j(\ell_i)}. \tag{5.7}$$

The updated mixing coefficients λ_j are then calculated in the usual way:

$$\lambda_j = \frac{1}{n} \sum_{i=1}^{n} \gamma_j(\ell_i). \tag{5.8}$$

The above E and M-steps are then iterated until convergence or a maximum number of iterations (10 in our experiments) are reached. Note that the above algorithm becomes numerically unstable when the observations are very near zero and one. Our implementation simply sidesteps this issue by bounding the observations in $[\epsilon, 1-\epsilon]$ instead of $[0,\, 1]$ ($\epsilon = 10^{-4}$ in our experiments).

Finally, we obtain the probability of a sample being clean or noisy through the posterior probability:

$$p(j \mid \ell_i) = \frac{p(j)\, p(\ell_i \mid j)}{p(\ell_i)}, \tag{5.9}$$

where $j = 0$ denote clean classes and $j = 1$ noisy.

The loss values used to fit the mixture model correspond to the standard cross-entropy loss computed over an additional forward pass (Figure 5.1) after every epoch. We have obtained very similar results with the loss used during training, but we decide to use the additional forward pass for simplicity when including the dynamic bootstrapping correction and *mixup* in the training loss function. Additionally, we have experimented with different update frequencies of the mixture distribution, every five epochs or every half epoch, and obtained marginal variations in the results (see Subsection 5.2.3). It might be of interest, however, to reduce the frequency in larger datasets to reduce the computational cost.

The initial method proposed in [10] leveraged the additional flexibility of the beta distribution to better fit skewed components of the loss distribution. Despite recent

experiments showing that, in some cases, GMM provides competitive results (see Section 5.2.3) BMM provides, overall, more stable results. Note that recent state-of-the-art in label noise [103, 136, 196] leverages the observations in this chapters and includes a loss-based method to select clean and noisy samples.

5.1.2 Dynamically bootstrapping predictions

The quality of the representations learned by a CNN heavily depends on the loss function used, particularly when training under label noise. The standard categorical cross-entropy brings competitive results when the dataset is clean, but fails to do so when labels are corrupted. This loss encourages the model to fit the corrupted labels and results in features that do not generalize outside the training set [216]. To account for this, [150] proposed the static hard bootstrapping loss function as a mechanism to reduce the influence of the corrupted labels in the training: a perceptual term in the loss helps to correct the training objective

$$\mathcal{L}_B = -\sum_{i=1}^{n} \left((1 - w_i)\, \mathbf{y}_i + w_i \mathbf{z}_i \right)^T \log\left(\mathbf{h}_i\right), \tag{5.10}$$

where w_i weights the model prediction $\mathbf{z}_i$ in the loss function. [10] use $w_i = 0.2, \forall i$. We refer to this approach as static hard bootstrapping (ST-H). Additionally, [150] proposed a static soft bootstrapping loss (ST-S), with $w_i = 0.05, \forall i$, that uses the predicted *softmax* probabilities $\mathbf{h}_i$ instead of the class prediction $\mathbf{z}_i$. Unfortunately, using a fixed weight for all samples does not prevent fitting the noisy ones (Table 5.1 in Subsection 5.2.2) and, more importantly, applying a small fixed weight w_i to the prediction $\mathbf{z}_i$, or probabilities $\mathbf{h}_i$, limits the correction of a hypothetical noisy label $\mathbf{y}_i$.

In this chapter we propose a dynamic bootstrapping strategy that leverages the modeling of the noise in the dataset to give individual weights to each sample. As initially proposed in [10] the dynamic hard and soft bootstrapping loss functions set w_i dynamically as $p(j = 1 \mid \ell_i)$, and the BMM model is estimated after each training

epoch using the cross-entropy loss for each sample ℓ_i. Therefore, clean samples rely on their ground-truth label $\mathbf{y}_i$ ($1 - w_i$ is large), while in noisy ones their loss is dominated by their class prediction $\mathbf{z}_i$ or their predicted probabilities $\mathbf{h_i}$ (w_i is large), respectively, for hard and soft alternatives. Note that in mature stages of training, the CNN model should provide a good estimation of the true class for noisy samples. Subsection 5.2.2 compares static and dynamic bootstrapping, showing that dynamic bootstrapping gives superior results.

5.1.3 *Mixup* as an effective regularization for label noise

Interpolation training as proposed in [218] has shown to be very valuable in label noise [218, 103, 110, 137]. As an initial adaptation to label noise, we (originally [10]) propose an interpolation-based bootstrapping loss function that leverages the regularization and data augmentation effects of *mixup* to reduce the harm of label noise in representation learning. Particularly, *mixup* (introduced in detail in Subsection 4.1.2 and equations (4.6) and (4.7)) provides a mechanism to combine clean and noisy samples, computing a more representative loss to guide the training process. Even when combining two noisy samples, the loss computed can still be useful as one of the noisy samples may (by chance) contain the true label of the other one. As for preventing overfitting to noisy samples, the fact that samples and their labels are mixed, favors learning structured data while hindering learning the unstructured noise.

After the publication of the method proposed in this Chapter [10], several other approaches have included some adaptation of the interpolation training proposed in [218], which corroborates the relevance of *mixup* as a powerful regularization strategy for label noise training. Moreover, as later explored in [136] and [137], *mixup* alone provides top performance in real-world datasets.

Mixup achieves robustness to label noise by appropriate combinations of training examples. Under high-levels of noise, mixing samples that both have incorrect labels is prevalent, which reduces the effectiveness of the method. We propose to fuse *mixup*

and our dynamic bootstrapping to implement a robust per-sample loss correction approach:

$$\mathcal{L}^* = -\delta \left[\left((1 - w_p)\, \mathbf{y}_p + w_p \mathbf{z}_p \right)^T \log\left(\mathbf{h}\right) \right] - (1 - \delta) \left[\left((1 - w_q)\, \mathbf{y}_q + w_q \mathbf{z}_q \right)^T \log\left(\mathbf{h}\right) \right], \tag{5.11}$$

where δ correspond to the strength of the *mixup* interpolation. The loss $\mathcal{L}^*$ defines the hard alternative, while the soft one $\mathcal{L}^*_{soft}$ can be easily defined by replacing $\mathbf{z}_p$ and $\mathbf{z}_q$ by $\mathbf{h}_p$ and $\mathbf{h}_q$. These hard and soft losses exploit *mixup*'s advantages while correcting the labels through dynamic bootstrapping, i.e. the weights w_p and w_q that control the confidence in the ground-truth labels and network predictions are inferred from our unsupervised noise model: $w_p = p(k = 1 \mid \ell_p)$ and $w_q = p(k = 1 \mid \ell_q)$. We compute $\mathbf{h}_p$, $\mathbf{z}_p$, $\mathbf{h}_q$ and $\mathbf{z}_q$ by doing an extra forward pass, as it is not straightforward to obtain the predictions for samples p and q from the mixed probabilities $\mathbf{h}$.

Ideally, the proposed loss $\mathcal{L}^*$ would lead to a better model by trusting in progressively better predictions during training. For high levels of label noise, however, the network predictions are unreliable and dynamic bootstrapping may not converge when combined with the complex signal that *mixup* provides. This is reasonable as under high levels of noise most of the samples are guided by the network's prediction in the bootstrapping loss function, encouraging the network to predict the same class to minimize the loss value. We apply the regularization term used in [175], which seeks to prevent the assignment of all samples to a single class, to overcome this issue:

$$R = \sum_{j=1}^{c} p_j \log\left(\frac{p_j}{\overline{h}_j}\right), \tag{5.12}$$

where p_j denotes the prior probability distribution for class j and $\overline{h}_j$ is the mean *softmax* probability of the model for class j across all samples in the dataset. Note that we assume a uniform distribution for the prior probabilities (i.e. $p_j = 1/c$), while approximating $\overline{h}_j$ using mini-batches as done in [175]. We add the term ηR to $\mathcal{L}^*$ (Eq. (5.11)) with η being the regularization coefficient (set to one in all the experiments). Subsection 5.2.3 presents the results of this approach and

Subsection 5.2.5 demonstrates its superior performance in comparison to the state-of-the-art.

To further increase the robustness of the approach in scenarios with higher levels of noise, the soft alternative $\mathcal{L}^*_{soft}$ overcomes the inconvenience of mixing noisy samples most of the time by providing a more precise guide. In this case we add a second regularization term, often used with soft labels [57], that encourages the network to minimize the entropy of the prediction:

$$R_H = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{c} h_i^j \log\left(h_i^j\right), \tag{5.13}$$

where h_i^j corresponds to the j class value of the *softmax* output $\mathbf{h}_i$ for the sample $\mathbf{x}_i$. This value is, as well, approximated using mini-batches (i.e. n is replaced by the mini-batch size).

5.2 Experiments and results

In this section, we empirically explore how the dynamic bootstrapping proposed in [10] addresses the challenges presented by label noise. Additionally, the other two approaches initially presented in [136] and [137] are studied. The experimental setup used in this chapter is introduced in Subsection 5.2.1. Experiments in Subsection 5.2.2 demonstrate the effectiveness of bootstrapping in label noise training and how the proposed dynamic bootstrapping better guides the training and exploits the potentially corrupted labels. In Subsection 5.2.3, we explore *mixup* in conjunction with dynamic bootstrapping. In Subsection 5.2.4, then, we evaluate the proposed approach in more extreme scenarios and propose adaptations to make it more robust. Finally, in Subsection 5.2.6 and 5.2.5, we compare the approach to state-of-the-art methods and explore the generalization to other datasets and across different noise distributions.

5.2.1 Experimental framework

Datasets and training setup

The main body of experiments in this chapter explores the behaviors of the proposed method on CIFAR-10 and CIFAR-100. We use a standard pre-processing for each image: normalized and augmented by random horizontal flipping and random crops after zero padding with four pixels on each side. The backbone model is a PreAct ResNet-18 [71] trained with SGD and momentum (0.9), weight decay of 10^{-4}, and batch size 128. The initial learning rate is 0.1 and is divided by 10 two or three times during the training depending on the configuration of the method: when training without *mixup*, the model trains for 120 epochs, the learning rate is reduced in epochs 30, 80, and 110, and the setup has a warm-up of 30 epochs, i.e. the bootstrapping starts at epoch 31; when training with *mixup* the model trains for 300 epochs, the learning rate is reduced in epochs 100 and 250, and warm-up for 105 epochs, i.e. bootstrapping starts at epoch 106. We present further experimentation in Subsection 5.2.6 on TinyImageNet (subset of ImageNet [42]) and Clothing1M [204] datasets that test the generality of our approach far from CIFAR data.

Experiments across all datasets share the same hyperparameter configuration and lead to consistent improvements over the state-of-the-art, demonstrating that the general approach does not require carefully tuned hyperparameters. Consequently, the results are likely to be suboptimal and could be improved with a label noise-free validation set, though this set is assumed to be unavailable in this research. Note that high learning rate values at early stages of the training are important to learn the structured data (mainly associated to clean samples) and to help separate the loss values between clean and noisy samples for a better BMM fit. Additionally, training more epochs leads to better performance, as *mixup* together with a high learning rate helps prevent fitting label noise.

The parameter estimation of the BMM consists of 10 EM iterations, but we ran M-DYR-H (80% of label noise, CIFAR-10) using 5 and 20 EM iterations, obtaining 87.4 and 87.2 for 5 iterations, and 86.9 and 86.3 for 20 iterations for *best* and *last*

epochs, suggesting that the method is relatively robust to this hyperparameter.

Dataset pre-processing: Label noise introduction

By pre-processing the datasets as described in Chapter 2 Subsection 2.3.2, we create different benchmarks for different noise levels. Initially, for the main body of experiments in this Chapter, random symmetric noise distributions, we follow [216, 218, 175] criterion: labels are swapped between all the available classes (i.e. the true label could be randomly maintained). Note that there is another popular label noise criterion for symmetric distributions [86, 194] in which the true label is not selected when performing random labeling. We also run our proposed approach under these conditions in Subsection 5.2.5 for comparison.

To further test the robustness of our model, we introduce asymmetric label noise in CIFAR-10 to simulate corruptions due to similarities between classes [141, 175]. Concretely we introduce the noise as in [213] by swapping the labels depending on the classes as follows: "truck" $\rightarrow$ "automobile", "bird" $\rightarrow$ "airplane", "deer" $\rightarrow$ "horse", and "cat" $\rightarrow$ "dog". This is a class-dependent label noise that is known as non-uniform or asymmetric noise and the noise level corresponds to the probability of a label being changed to the corresponding noisy class. This scenario is further studied together with more complex and realistic distributions in [136] and [137], which also evaluate the method described in this chapter and proposed in [10]. Note that this thesis also presents and discusses results of this research in Section 5.3.

5.2.2 Static and dynamic loss correction: dynamic bootstrapping

Table 5.1 compares static (ST) and dynamic (DY) bootstrapping in CIFAR-10 with different levels of symmetric noise. When evaluating the best performance of the different approaches, ST achieves comparable results to DY (except for 80% noise where DY is much better), but the model after the last epoch (last) of DY significantly outperforms ST. This shows that the dynamic weighting of each sample contributes

Table 5.1: Validation accuracy on CIFAR-10 for static bootstrapping and the proposed dynamic bootstrapping. Key: CE (cross-entropy loss), ST (static bootstrapping), DY (dynamic bootstrapping), S (soft), and H (hard). Bold indicates best performance.

Alg./Noise level (%)		0	20	50	80
CE	Best	93.8	**89.7**	**84.8**	67.8
	Last	93.7	81.8	55.9	25.3
ST-S	Best	**93.9**	**89.7**	84.8	67.8
	Last	**93.9**	81.7	55.9	24.8
ST-H	Best	93.8	**89.7**	**84.8**	68.0
	Last	93.8	81.4	56.4	25.7
DY-S	Best	93.6	**89.7**	**84.8**	67.8
	Last	93.4	83.3	57.0	27.8
DY-H	Best	93.3	**89.7**	**84.8**	**71.7**
	Last	92.9	**83.4**	**65.0**	**64.2**

to reducing overfitting to label corruptions. The improvements are particularly remarkable for 80% of label noise (from 25.7% of ST-H to 64.2 of DY-H). Table 5.1 also shows that hard (H) gives superior performance to soft (S) bootstrapping, which is consistent with the findings of the original paper [150]. The overall results demonstrate that applying per-sample weights (DY) benefits training by allowing full correction of noisy labels.

5.2.3 *mixup* in conjunction with dynamic bootstrap

The proposed dynamic hard bootstrapping exhibits better performance than the state-of-the-art static version [150]. It is, however, not better than the performance of *mixup* data augmentation, which exhibits excellent robustness to label noise (M in Table 5.2). The fusion approach from Eq. (5.11) (M-DYR-H) and its soft alternative (M-DYR-S), which combines the per-sample weighting of dynamic bootstrapping and robustness to fitting noise labels of *mixup*, achieves a remarkable improvement in accuracy under high noise levels. Table 5.2 reports outstanding accuracy for 80% of label noise, a case where we improve upon *mixup* [218] in *best* (*last*) accuracy of 71.6 (46.7) in CIFAR-10 and 30.8 (17.6) in CIFAR-100 to 86.8 (86.6) and 48.2 (47.2) using the hard alternative (M-DYR-H). Note that the soft alternative alone (M-DYR-S) fails to surpass these results, but provides state-of-the-art across most noise levels

Table 5.2: Validation accuracy on CIFAR-10 (top) and CIFAR-100 (bottom) for joint *mixup* and boot strapping. Key: CE (cross-entropy), M (*mixup*), DYR (dynamic bootstrapping + regularization from Eq. (5.12)), S (soft), H (hard), and S2 (soft + regularization from Eq. (5.13)). Bold indicates best performance.

Alg./Noise level (%)		0	20	50	80
CE	Best	94.7	86.8	79.8	63.3
	Last	94.6	82.9	58.4	26.3
M [218]	Best	**95.3**	**95.6**	87.1	71.6
	Last	**95.2**	92.3	77.6	46.7
M-DYR-S	Best	93.3	93.5	89.7	77.3
	Last	93.0	93.1	89.3	74.1
M-DYR-H	Best	93.6	94.0	92.0	**86.8**
	Last	93.4	93.8	91.9	**86.6**
M-DYR-S2	Best	93.5	94.6	**93.2**	85.6
	Last	93.2	**94.4**	**93.0**	85.3

Alg./Noise level (%)		0	20	50	80
CE	Best	**76.1**	62.0	46.6	19.9
	Last	**75.9**	62.0	37.7	8.9
M [218]	Best	74.8	67.8	57.3	30.8
	Last	74.4	66.0	46.6	17.6
M-DYR-S	Best	71.9	67.9	61.7	38.8
	Last	67.4	67.5	58.9	34.0
M-DYR-H	Best	70.3	68.7	61.7	48.2
	Last	66.2	68.5	58.8	47.6
M-DYR-S2	Best	72.1	**72.5**	**67.6**	**50.5**
	Last	71.3	**72.1**	**67.6**	**50.5**

in CIFAR-10 and CIFAR-100 when combined with the regularization described in Eq. (5.13) (M-DYR-S2), which demonstrates the efficiency of encouraging low entropy predictions to guide the training (as discussed in Chapter 4).

It is important to highlight that we achieve quite similar *best* and *last* performance for all levels of label noise in CIFAR datasets, indicating that the proposed method is robust to varying noise levels. Figure 5.3 shows uniform manifold approximation and projection (UMAP) embeddings [119] of the 512 features in the penultimate fully-connected layer of PreAct ResNet-18 trained using our method and compares them with those found using cross-entropy and *mixup*. The separation among classes appears visually more distinct using the proposed objective. Additionally, we experimented on CIFAR-10 fitting a BMM per class aiming at obtaining a more

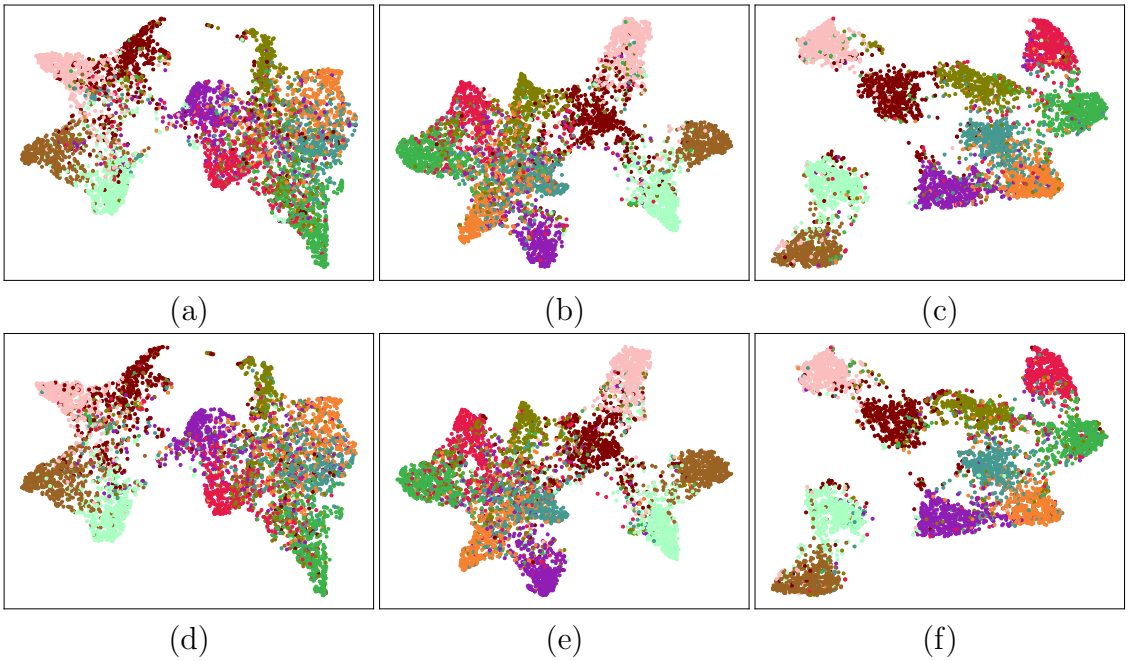


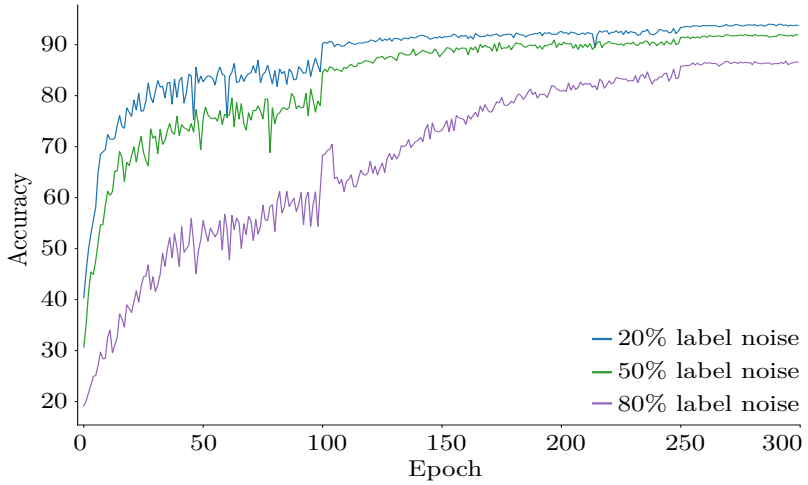
Figure 5.3: UMAP [119] embeddings for training (top) with 80% of label noise and validation (bottom) on CIFAR-10 with (a)(d) cross-entropy loss from Eq. (5.1), (b)(e) *mixup* [218] and (c)(f) our proposed M-DYR-H.

tailored fit to the distribution, which provided very similar results.

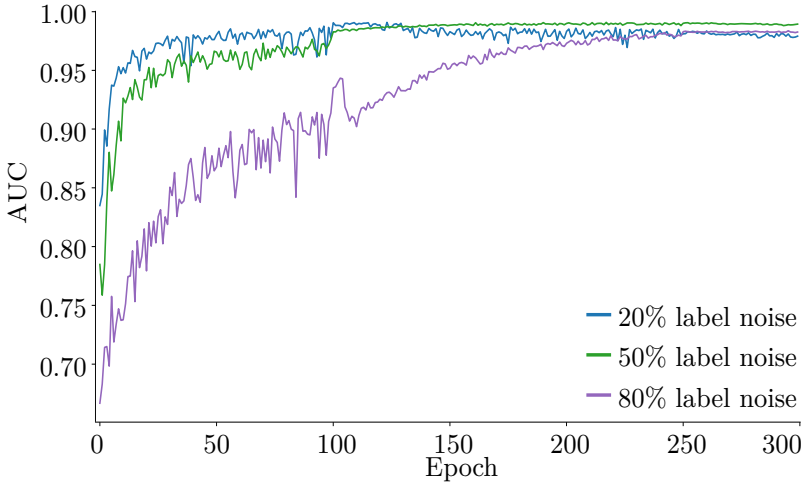
Further exploration of the noise modeling approach: BMM configuration

The proposed approach re-estimates the BMM parameters every epoch once the loss correction begins (i.e. there is an initial warm-up as noted in Subsection 5.2.1 with no loss correction) by computing the cross-entropy loss from a forward pass with the original (potentially noisy) labels. We also explored different estimation periods (every 5 and 0.5 epochs) for M-DYR-H in CIFAR-10 and 80% of label noise and observed similar results. While the original configuration presented in Figure 5.4 (a) reaches 86.8 and 86.6 for *best* and *last*, every 5 epochs leads to 86.9 and 86.8, and every 0.5 to 88.0 and 87.5.

We expect a drop in performance in carefully annotated datasets that resemble the 0% noise scenario in CIFAR datasets: where there is only one component in the loss distribution to fit the BMM. Despite classifying almost all samples as clean, estimation errors occur which lead to a reliance on the sometimes incorrect network prediction instead of the true clean label. Nevertheless, for 20% noise, the approach



(a)



(b)

Figure 5.4: M-DYR-H results on CIFAR-10 for (a) image classification and (b) clean/noisy classification of the BMM.

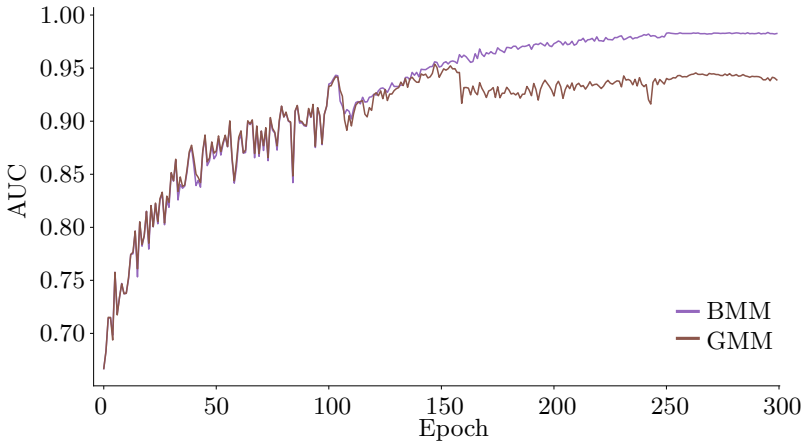


Figure 5.5: M-DYR-H results on CIFAR-10: comparison of GMM and BMM for clean/noisy classification with 80% label noise.

proposed in this chapter outperforms the compared state-of-the-art at the end of the training, demonstrating improved robustness for low noise levels.

Additionally, Figure 5.4 (b) shows the clean/noisy classification capabilities of the BMM in terms of Area Under the Curve (AUC) evolution during training, demonstrating that performance and robustness are consistent across noise levels. In particular, the experiment on CIFAR-10 with M-DYR-H exceeds 0.98 AUC for 20, 50, and 80% label noise. AUC increases during training and increases faster for lower noise levels, showing increasingly better clean/noisy discrimination related to consistent BMM predictions over time.

Effect of BMM classification accuracy on image classification accuracy

BMM prediction accuracy is essential for high image classification accuracy, as demonstrated by the tendency for both image classification and BMM accuracy to increase together in Figure 5.4 (a) and (b), especially for higher noise levels. Figure 5.5 further verifies this relationship by comparing the BMM with a GMM (Gaussian Mixture Model) on CIFAR-10 with M-DYR-H and 80% label noise. In this scenario, with $\alpha = 32$, the GMM gives both less accurate clean/noisy discrimination and worse image classification results (clean/noisy AUC drops from 0.98 to 0.94, while image classification accuracy drops from 82.0 to 80.6). Note that further experiments have shown that GMM performs on par with BMM in CIFAR-100 and provides slightly better convergence in higher levels of noise, particularly when decreasing the interpolation strength applied in *mixup*. Table 5.3 provides more details on these experiments and shows the dependence between the effectiveness of the noise detection and the regularization strength imposed by the α parameter of *mixup*.

As an additional ablation study to verify the contribution of the BMM estimations, we remove the BMM and assign fixed weights in the bootstrapping loss (0.8 to the ground truth label and 0.2 to network prediction, keeping *mixup* for robustness). This leads to a drop from 86.6 for M-DYR-H to 74.6 in the last epoch (80% of label

Table 5.3: Validation accuracy of M-DYR-H (*best*/*last*): comparison of BMM and GMM. Bold indicates best performance.

	CIFAR-10		CIFAR-100	
	BMM	GMM	BMM	GMM
Noise level	$\alpha = 1$			
0 %	**95.07**/79.63	95.01/**87.19**	**76.32/75.66**	76.06/67.97
40 %	**91.80/91.48**	88.91/51.46	64.61/63.81	64.00/61.89
80 %	80.41/**67.79**	**82.66**/63.66	27.54/24.49	**42.41/42.41**
Noise level	$\alpha = 32$			
0 %	**94.03/93.34**	92.96/74.43	**70.59/68.61**	70.59/33.79
40 %	**92.32/91.81**	90.72/88.20	**64.27/62.53**	63.07/30.51
80 %	**83.44/82.00**	81.67/80.65	**47.69/45.02**	44.97/38.88

Table 5.4: Validation accuracy of M-DYR-H in CIFAR-10 and symmetric noise (*best*/*last*) for different α values. Bold indicates best performance.

Noise level/α	0.1	0.5	1	2	4	8	32
0 %	95.02/**94.67**	**95.43**/89.79	95.34/89.95	95.07/87.82	94.41/90.85	94.27/91.82	94.14/94.14
40 %	82.46/70.68	87.33/75.24	90.11/84.82	91.58/88.30	92.90/92.43	93.32/93.10	**93.41/93.29**
80 %	67.32/30.59	70.64/40.53	81.98/54.33	86.48/79.23	88.58/88.11	**89.28/88.72**	86.68/85.95

noise on CIFAR-10).

Further exploration of *mixup* data augmentation configuration

Table 5.4 compares different values for the α parameter in *mixup*, which defines the shape of the beta distribution that yields the strength of the interpolation in every iteration: higher values of α lead to similar contributions from both samples while smaller values lead to the dominance of one of the samples. These results show that higher values of α provide more robust results for higher levels of noise, which supports the claim that stronger regularization strategies help training CNNs in label noise scenarios, particularly when the noise levels increase. We keep $\alpha = 32$ unless otherwise stated.

5.2.4 Extreme cases of label noise

Table 5.5 presents an experimental exploration of model convergence under extreme label noise conditions, showing that the proposed approach M-DYR-H fails to converge in CIFAR-10 with 90% label noise (as also do the proposed alternatives

Table 5.5: Validation accuracy on CIFAR-10 (top) and CIFAR-100 (bottom) with extreme label noise. Key: M (*mixup*), MD (dynamic *mixup*), DYR (dynamic bootstrapping + reg. from Eq. (5.12)), H (hard), SH (soft to hard), and S2 (soft + regularization from Eq. (5.13). (*) denotes that we have run the algorithm. Bold indicates best performance.

Alg./Noise level (%)		70	80	85	90
M-DYR-H	Best	89.6	**86.8**	71.6	40.8
	Last	89.6	**86.6**	71.4	9.9
MD-DYR-H	Best	86.6	83.2	**79.4**	56.7
	Last	85.2	80.5	**77.3**	50.0
MD-DYR-SH	Best	84.6	82.4	79.1	**69.1**
	Last	80.8	77.8	73.9	**68.7**
M-DYR-S2	Best	**91.2**	85.6	58.3	40.8
	Last	**91.0**	85.3	39.7	10.6
Alg./Noise level (%)		70	80	85	90
M-DYR-H	Best	54.4	48.2	29.9	12.5
	Last	52.5	47.6	29.4	8.6
MD-DYR-H	Best	54.4	47.7	19.8	13.5
	Last	50.8	41.7	8.3	3.9
MD-DYR-SH	Best	53.1	41.6	28.8	**24.3**
	Last	47.7	35.4	24.4	**20.5**
M-DYR-S2	Best	**61.3**	**50.5**	**32.3**	12.5
	Last	**60.9**	**50.5**	**32.1**	6.8

M-DYR-S and M-DYR-S2). Minor modifications to achieve convergence are proposed in this subsection.

When clean and noisy samples are combined by *mixup* they are given the same importance of approximately $\delta = 0.5$ (as $\alpha = \beta = 32$. While noisy samples benefit from mixing with clean ones, clean samples are contaminated by noisy ones, whose training objective is incorrectly modified. We propose a dynamic *mixup* strategy in the input that uses a different δ for each sample to reduce the contribution of noisy samples when they are mixed with clean ones:

$$\mathbf{x} = \left(\frac{\delta_p}{\delta_p + \delta_q}\right)\mathbf{x}_p + \left(\frac{\delta_q}{\delta_p + \delta_q}\right)\mathbf{x}_q, \tag{5.14}$$

where $\delta_p = p(k = 0 \mid \ell_p)$ and $\delta_q = p(k = 0 \mid \ell_q)$, i.e. we use the noise probability from our BMM to guide *mixup* in the input. Note that for clean-clean and noisy-noisy cases, the behavior remains similar to *mixup* with $\alpha = \beta = 32$, which leads to $\delta \approx 0.5$

(i.e. $\delta_p \approx \delta_q \Rightarrow \delta_p/(\delta_p + \delta_q) \approx 0.5$).

This configuration simplifies the input to the network when mixing a sample whose label is potentially useless (or even harmful) while retaining the strengths of *mixup* for clean-clean and noisy-noisy combinations. This is used with the original *mixup* strategy (Eq. (5.11)) to benefit from the regularization that an additional label provides. Table 5.5 presents the results of this approach (MD-DYR-H), which exhibits more stable convergence for 90% label noise in both datasets. Note that this approach assigns higher weights to the clean samples, hence reducing the contribution of noisy samples in the training. This resembles the approaches to curriculum learning or importance sampling introduced in Chapter 2, where different samples have different relevance through the training.

Table 5.2 reports that hard bootstrapping works better than the soft alternative. Unfortunately, hard bootstrapping under high levels of label noise causes large variations in the loss that lead to drops in performance. To ameliorate such instabilities, we propose a decreasing *softmax* technique [189] to progressively move from a soft to a hard dynamic bootstrapping. This is implemented by modifying the *softmax* temperature T in:

$$h_{ij} = \frac{\exp(s_{ij}/T)}{\sum_{k=1}^{n} \exp(s_{ik}/T)}, \tag{5.15}$$

where s_{ij} denotes the score obtained in the last layer of the CNN model for the element j, i.e. class j, in the prediction the of a sample $\mathbf{x}_i$. By default $T = 1$ gives the soft alternative of Eq. (5.11). To move from soft to hard bootstrapping we linearly reduce the temperature for $\mathbf{h}_p$ and $\mathbf{h}_q$ until we reach a final temperature in a certain epoch ($T = 0.001$ and epoch 200 in our experiments). We experimented with linear, logarithmic, tanh, and step-down temperature decays with similar results. This decreasing *softmax* MD-DYR-SH obtains much improved accuracy for 90% of label noise (69.1 for CIFAR-10 and 24.3 for CIFAR-100), while slightly decreasing accuracy compared to M-DYR-H and MD-DYR-H at lower noise levels. We significantly outperform previous state-of-the-art for 90% of label noise, which is 58.3% and 58.0% for *best* and *last* validation accuracies (reported in [175] with a PreAct ResNet-32 on

CIFAR-10). The training process is slightly modified to introduce dynamic *mixup* (epoch 106) before bootstrapping (epoch 111) for MD-DYR-H and MD-DYR-SH. Note that, despite M-DYR-S2 providing better results than MD-DYR-SH across noise levels, it still fails to converge to a competitive accuracy in extreme levels of noise (90%). This could be due to weak guidance in later stages of the training given by the soft-labels.

5.2.5 Comparison to the state-of-the-art

This subsection compares the proposed dynamic bootstrapping loss function with relevant approaches from the literature. The work presented in [136, 137], however, further explores this and other approaches in different label noise distributions. It is also of particular interest to this thesis the approach from [103], denoted as DM, that combines the loss modeling proposed in this chapter with the co-training technique proposed in [146] to provide very competitive results. Note that despite being simpler approaches, in terms of hyperparameter tuning and training epochs, the methods proposed in [136, 137] (in close collaboration with the development of this thesis), outperformed DM in most of the benchmarks, particularly in real-world datasets (Subsection 5.3 further discusses these results).

Table 5.6 compares related works for different levels of label noise using a common architecture and the 300 epochs training scheme (see Subsection 5.2.1). We introduce bootstrapping in epoch 105 for ST_H [150] and for the proposed methods, estimate the T matrix of F [141] in epoch 75 (as done in [73], and use the configuration reported in [218] for *mixup*. We outperform the related work in the presence of label noise, obtaining improvements for high levels of noise (80% and 90%) where the compared approaches do not learn as well from the noisy samples (see *best* accuracy) and do not prevent fitting noisy labels (see *last* accuracy).

To allow for quantitative comparison we evaluate, in Table 5.7, the proposed approach under symmetric noise introduced as in [86, 115, 151, 194] (as noted in Subsection 5.2.1), where the true label is excluded when randomly swapping labels.

Table 5.6: Comparison with the state-of-the-art in terms of validation accuracy on CIFAR-10 (top) and CIFAR-100 (bottom). Key: ST-H (static hard bootstrapping), F (forward), M (*mixup*), MD (dynamic *mixup*), DYR (dynamic bootstrapping + reg. from Eq. (5.12)), H (hard), SH (soft to hard), and S2 (soft + regularization from Eq. (5.13). (*) denotes that we have run the algorithm. Bold indicates best performance.

Alg./Noise level (%)		0	20	50	80	90
ST-H [150]*	Best	94.7	86.8	79.8	63.3	42.9
	Last	94.6	82.9	58.4	26.8	17.0
F [141]*	Best	94.7	86.8	79.8	63.3	42.9
	Last	94.6	83.1	59.4	26.2	18.8
M [218]*	Best	**95.3**	**95.6**	87.1	71.6	52.2
	Last	**95.2**	92.3	77.6	46.7	43.9
M-DYR-H	Best	93.6	94.0	92.0	**86.8**	40.8
	Last	93.4	93.8	91.9	**86.6**	9.9
MD-DYR-SH	Best	93.6	93.8	90.6	82.4	**69.1**
	Last	92.7	93.6	90.3	77.8	**68.7**
M-DYR-S2	Best	93.5	94.6	**93.2**	85.6	40.8
	Last	93.2	**94.4**	**93.0**	85.3	10.6

Alg./Noise level (%)		0	20	50	80	90
ST-H [150]*	Best	**76.1**	62.1	46.6	19.9	10.2
	Last	**75.9**	62.0	37.9	8.9	3.8
F [141]*	Best	75.4	61.5	46.6	19.9	10.2
	Last	75.2	61.4	37.3	9.0	3.4
M [218]*	Best	74.8	67.8	57.3	30.8	14.6
	Last	74.4	66.0	46.6	17.6	8.1
M-DYR-H	Best	70.3	68.7	61.7	48.2	12.5
	Last	66.2	68.5	58.8	47.6	8.6
MD-DYR-SH	Best	73.3	**73.9**	66.1	41.6	**24.3**
	Last	71.3	**73.4**	65.4	35.4	**20.5**
M-DYR-S2	Best	72.1	72.5	**67.6**	**50.5**	12.5
	Last	71.3	72.1	**67.6**	**50.5**	6.8

In this case, label noise is defined as the percentage of incorrect labels instead of random ones (i.e. the criterion followed in previous experiments). The proposed method outperforms all related work in CIFAR-10 and CIFAR-100 with MD-DYR-SH, while the results for M-DYR-H are slightly below those of [86] for low label noise levels in CIFAR-100. Nevertheless, these results should be interpreted with care due to the different architectures employed and the use of sets of clean data during training in [86] and [151].

Table 5.7: Comparison with the state-of-the-art in terms of validation accuracy on CIFAR-10 (top) and CIFAR-100 (bottom). Key: M (*mixup*), MD (dynamic *mixup*), DYR (dynamic bootstrapping + reg. from Eq. (5.12)), H (hard), SH (soft to hard), WRN (Wide ResNet), PRN (PreActivation ResNet, and GCNN (Generic CNN). Bold indicates best performance.

Algorithm	Architecture	Noise level (%)			
		20	40	60	80
MentorNet [86]	WRN-101	92.0	89.0	-	49.0
DimDriven [115]	GCNN-12	85.1	83.4	72.8	-
L2-Reweight [151]	WRN-28	-	86.9	-	-
Iterative learning [194]	GCNN-7	81.4	78.2	-	-
M-DYR-H	PRN-18	**94.0**	**92.8**	**90.3**	46.3
MD-DYR-SH	PRN-18	93.8	92.3	86.1	**74.1**

Algorithm	Architecture	Noise level (%)			
		20	40	60	80
MentorNet [86]	WRN-101	73.0	68.0	-	35.0
DimDriven [115]	RN-44	62.2	52.0	42.3	-
L2-Reweight [151]	WRN-28	-	61.3	-	-
M-DYR-H	PRN-18	70.0	64.4	58.1	**45.5**
MD-DYR-SH	PRN-18	**73.7**	**70.1**	**59.5**	39.5

5.2.6 Generalization: larger images and realistic noise distributions

Other datasets

Table 5.8 shows the results of the proposed approaches M-DYR-H, MD-DYR-SH, and M-DYR-S2 compared to *mixup* [218] on TinyImageNet to demonstrate that our approach is useful far from CIFAR data. The proposed approach outperforms [218] for different levels of label noise, obtaining consistent results with the CIFAR experiments. Note that we use the same network, hyperparameters, and learning rate policy as with CIFAR.

For Clothing1M we followed a similar network and procedure as [175] with ImageNet pre-trained weights and ResNet-50, obtaining over 71% test accuracy, which falls short of the state-of-the-art (72.23% [175]). We found that fine-tuning a pre-trained network for one epoch, as done in [175], easily fits label noise, limiting our unsupervised label noise model. We believe this occurs due to the structured nature of the noise in this dataset, the small learning rate, and the large amounts of

Table 5.8: Comparison of test accuracy on TinyImageNet. Key: M (*mixup*) , DYR (dynamic bootstrapping + reg. from Eq. (5.12)), H (hard), and SH (soft to hard). (*) denotes that we have run the algorithm. Bold indicates best performance.

Alg./Noise level (%)		20	50	80
M [218]*	Best	53.2	41.7	18.9
	Last	49.4	31.1	8.7
M-DYR-H	Best	51.8	44.4	18.3
	Last	51.6	43.6	17.7
MD-DYR-SH	Best	**60.0**	**50.4**	**24.4**
	Last	**59.8**	**50.0**	**19.6**

data compared to the CIFAR experiments. Training with cross-entropy alone gives test accuracy over 69%, suggesting that the configurations used might be suboptimal.

Other noise distributions

We evaluate our approach in a more realistic label noise distribution by training our method on CIFAR-10 with non-uniform (asymmetric) noise generated as described in Subsection 5.2.1, where label flips are concentrated in classes sharing similar visual patterns with the true class. Table 5.9 shows that the proposed approaches M-DYR-H and M-DYR-S2 are a robust methodology for training in asymmetric label noise scenarios, especially when the levels of noise are higher: for all the levels of label noise they perform on pair with the best performing method Joint Optimization [175]. The experimental section of [136, 137] further explores this approach (among others) in more complex and realistic scenarios (including OOD samples). As a highlight of the experiments on these scenarios, Table 5.10, extracted from [136], compares the approach discussed in this chapter with recent approaches in WebVision [105], a dataset with annotations obtained directly from the web (see Chapter 2 for more details on this dataset). Note that M-DYR-H outperforms the simpler approaches (CE, FW, R) but fall short when compared to those that include strong regularization techniques (M and DRPL).

Table 5.9: Comparison of the proposed approach M-DYR-H and M-DYR-S2 with the state-of-the-art in terms of validation accuracy on CIFAR-10 with asymmetric noise. Key: CE (cross-entropy), M (*mixup*), F (Forward), J.O. (Joint Optimization). Bold indicates best performance.

Alg./Noise level (%)		10	20	30	40
CE*?	Best	92.1	90.5	89.5	86.7
	Last	90.1	85.5	80.8	74.5
M [218]*	Best	**94.5**	93.0	90.4	87.3
	Last	**94.2**	93.0	89.7	84.6
F [141]*	Best	91.3	88.1	87.5	84.7
	Last	91.2	86.2	82.6	76.4
J.O. [175]*	Best	93.8	**93.4**	**93.2**	**91.5**
	Last	93.6	**93.4**	**93.1**	**91.5**
M-DYR-H	Best	92.5	92.1	92.0	91.2
	Last	92.0	88.3	91.7	91.0
M-DYR-S2	Best	93.3	93.3	92.6	91.3
	Last	93.2	93.2	92.6	91.3

Table 5.10: Top-1 accuracy in first 50 classes of WebVision. Results originally reported in [136].

	CE	F [141]	R [175]	M [218]	M-DYR-H	P[213]	DM[103]	DRPL[136]
Best	73.88	74.68	76.52	80.76	79.68	79.96	78.16	**82.08**
Last	73.76	74.32	76.24	79.96	79.56	79.44	76.84	**82.00**

5.3 Synthetic and real-world noise distributions

In this section, we present additional work initially discussed in [136, 137] and provides further insights into the dynamics of training in the presence of label noise. Concretely, these two works explored the effects that different noise distributions have on the training of CNN and proposed robust techniques to palliate the harmful effect of noisy labels have in representation learning.

The main difference between symmetric, asymmetric, and real-world noise distributions is the dependence of the incorrect labels on the corresponding images. This dependence directly affects the learning process and interferes with representation learning at different levels. As shown in [136], the differences between these noise distributions can be seen in the quality of the features learned and in the label noise detection capabilities of the model. The first type of noise distribution, symmetric noise (or uniform), considerably hinders feature quality when not accounted for, but

allows for an easy identification through the inspection of individual loss values; we believe its presence in real-world datasets is small. The second type, asymmetric noise (or non-uniform), shows a smaller effect on feature quality but is significantly more challenging to identify and it requires more complex approaches rather than just evaluating the loss value; this type of noise seems to be more abundant in real-world noise.

These two types of noise distribution simulate two levels of dependence between the noisy label and the image but do not account for an important factor in real-world datasets: out-of-distributions samples. However, recently proposed benchmarks, ImageNet-32 [136] and Mini-ImageNet-cwln [85], allow for an exploration of different levels of OOD samples. This section introduces initial findings on these scenarios, further explored in [136, 137], which constitute a pivotal part of the future research work described in Chapter 7.

Subsection 5.3.2 addresses how noise modeling stages are affected by the different noise distributions and provides intuitions on the organization of the feature space when learning under label nose. Subsection 5.3.1 explores how the presence of label noise corrupts CNN features and provides visualizations of the memorization effect when training under these conditions.

5.3.1 Noise detection

Figure 5.6 illustrates the influence of the noise distribution in the quality of the loss values as a feature to discriminate between clean and noisy samples. Figure 5.6 (left) shows the loss values of a model trained in the presence of symmetric noise, the separation between the values corresponding to clean (blue) and noisy samples (red) reflects the accuracy with which these two subsets can be separated (i.e. label noise detection). Figure 5.6 (right) shows how the loss values of clean and noisy samples overlap when the noise follows an asymmetric distribution; in this case, these values result in a less accurate noise detection stage. To deal with this, [136] proposes a relabeling technique that helps separate the loss values from noisy and

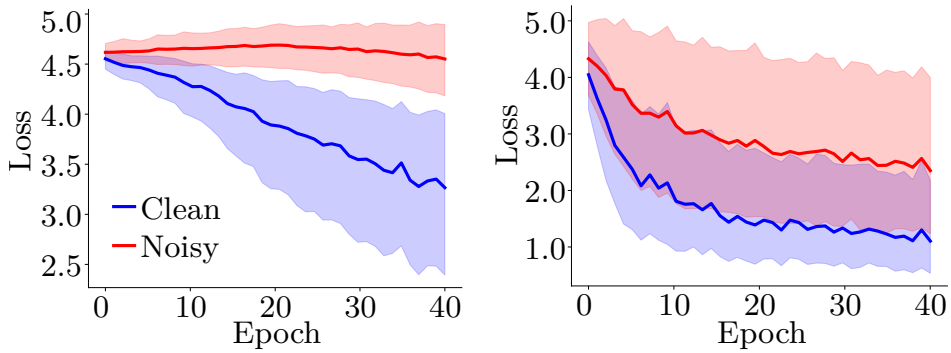


Figure 5.6: Images obtained from [136] that illustrate loss values for clean (blue) and noisy (red) samples for different label noise distributions in 100 classes of ImageNet-32: 80% symmetric (left) and asymmetric (right) in-distribution noise. Training: 40 epochs with a PreAct ResNet-18 [71] with learning rate of 0.1 and cross-entropy loss.

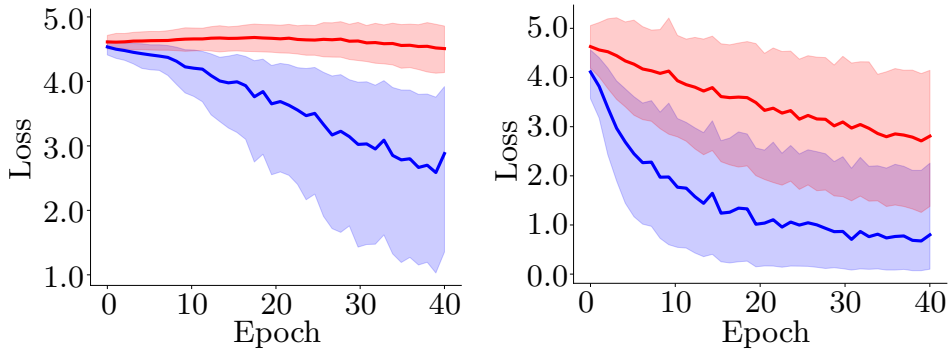


Figure 5.7: Images obtained from [136] that illustrate loss values for clean (blue) and noisy (red) samples for different label noise distributions in 100 classes of ImageNet-32: 80% symmetric (left) and asymmetric (right) out-of-distribution noise. Training: 40 epochs with a PreAct ResNet-18 [71] with learning rate of 0.1 and cross-entropy loss.

clean samples, and [137] proposes a measure of label discrepancy between a sample and its neighboring samples in the feature space. Both strategies target the scenario where the noise has some structure (i.e. there is a dependence between corrupted labels and images) and the individual sample loss values fail to discriminate between clean and noisy samples. Additionally, to illustrate the effect of more realistic noise distributions, Figure 5.7 shows loss values when the model is trained in the presence of a noise distribution where the noisy images do not belong to the distribution of labels in the testing set, i.e. OOD label noise.

To gain a better understanding of the feature space learned in the presence of label noise, it is particularly interesting the noise detection approach proposed in [137], since it is based on the nearest-neighbor graph of the features of the samples

extracted by a trained model. The basic assumption behind it is that noisy samples are allocated in heterogeneous regions of the feature space, meaning that they are surrounded by samples with a high variety of labels. However, clean samples are in homogeneous regions, among samples from a particular label. This homogeneity of labels is what the metric proposed in [137] evaluates to discriminate clean and noisy samples. Additionally, to improve noise detection, the labels used are consensus labels obtained from a pool of the nearest neighbors of each sample, i.e. all samples are relabeled, before the noise detection, with the most common label in a set of the nearest neighbors. Intuitively, this approach searches for label consistency across neighbors two "hops" from the current sample: i.e. the nearest neighbor of the nearest neighbor.

To further explore the landscape of the feature space in search of noisy samples, we explore graph diffusion mechanisms, as in [4, 81], to evaluate the neighborhood of each sample through the structure of a nearest-neighbor graph. Table 5.11 shows that graph diffusion (*Diffusion-p* and *Diffusion-$\hat{p}$*) and the label discrepancy measure proposed in [137] (k-NN-p and k-NN-$\hat{p}$) result in similar noise detection results (especially in CIFAR-100). To analyze the problem independently from the training method, Table 5.11 reports AUC score of label noise detection using the features learned after training a ResNet-18 with self-supervied strategies (i-Mix [102] for CIFAR-100 and SimCLR for mini-ImageNet-cwln [29]). k-NN-p corresponds to the label noise detection used in [137] that evaluates the discrepancy of the current label with that of the nearest neighbors, k-NN-$\hat{p}$ follows the same strategy, but first corrects the labels with the majority label in the neighborhood. Similarly, *Diffusion-p* and *Diffusion-$\hat{p}$* measure the discrepancy of the current label with the probability distribution obtained after the diffusion of the current label, *Diffusion-p*, or the corrected label, *Diffusion-$\hat{p}$*. The results suggest that considering further neighbors (k-NN-$\hat{p}$) is beneficial in symmetric noise for CIFAR-100, but harmful in the real-world noise from mini-ImageNet-cwln, Considering the nearest neighbor only (k-NN-p), however, results in opposite results. This suggest that the label of farther neighbors is

Table 5.11: AUC scores for noise detection strategies in 40% asymmetric noise in CIFAR-100 and 40% of web label noise in mini-ImageNet-cwnl ("red" noise as proposed in [85]). Bold indicates best performance.

	CIFAR-100	mini-ImageNet-cwln
k-NN-p	0.61	**0.75**
k-NN-$\hat{p}$	**0.85**	0.72
Diffusion-p	0.58	0.70
Diffusion-$\hat{p}$	**0.85**	0.73

informative in very controlled scenarios without OOD samples (asymmetric noise in CIFAR-100), but hinders noise detection when the noise distribution has a stronger instance dependent structure and contains OOD samples (mini-ImageNet-cwln). These experiments motivate further exploration of CNN training in the presence of real-world noise distributions: Chapter 7 includes this as one of the main future lies of work.

5.3.2 Training under label noise

The effect of different noise distributions in the representations learned by CNN provides relevant insights that help better understanding label noise. Results from [136, 137] show that similar levels of noise are less harmful when the distribution of noisy labels is asymmetric rather than symmetric. This is observable in the experiments on ImageNet-32/64 in [136] for uniform and non-uniform noise (symmetric and asymmetric) and in [137] for the experiments in CIFAR-100. Particularly, a CNN trained without addressing label noise, i.e. CE, under 40% of symmetric noise in CIFAR-100 reaches 42.92 and 44.46 under asymmetric noise [136]. The strategy to introduce the synthetic noise in CIFAR-10, however, does not allow for direct comparison between similar levels of symmetric and asymetric noise: in the case of asymmetric noise, only some classes are corrupted, resulting in a considerably lower number of corrupted samples across the dataset. Similar results are reported for ImageNet-32/64, showing higher average accuracy across noise levels when the noise corruptions follow a symmetric distribution. We suggest that this stems from

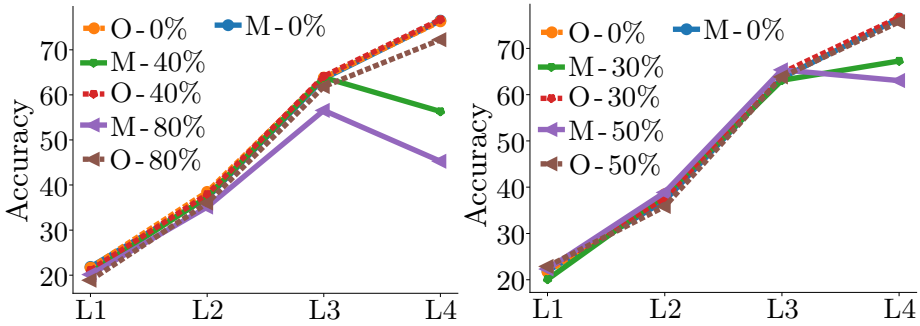


Figure 5.8: ImageNet-64 linear probes originally presented in [136]. A linear classifier is trained using CNN features at different depths. Noise: in-distribution symmetric (left) and asymmetric (right). Key: M: *mixup*. O: Ours.

training with labels that encode some structure of the dataset (asymmetric) against training with completely random labels that do not contain useful information of the samples (symmetric noise). The former scenario allows the CNN to leverage the structure to learn features with some degree of generalization, while the latter requires the CNN to allocate resources to pure memorization of labels that will not generalize to the test set.

In this line of understanding the role of label noise in CNN training, Ortego *et al.* [136] provide two more insightful observations of the corruption of CNN layers when trained under these conditions. First, Figure 5.8 shows that label noise degrades the quality of the representations learned by final layers in the model while maintaining it in earlier: for deeper layers (L1 and L2 in the image) the performance of different approaches for different levels of noise reach similar accuracy, but for shallower layers (L3 and L4) the accuracy of models trained under higher levels of noise drops with respect to the models trained without label noise. These results correspond to linear classifiers trained on top of the features learned by the CNN under symmetric (left) and asymmetric (right) noise in ImageNet-64 and tested on unseen data (see [136] for the details in the usage of linear proves to evaluate the utility of learned features). Similar observations were obtained when training in the presence of out-of-distribution samples. Second, Figure 5.9 provides the Class Activation Maps [226] (CAM) for the true class and the predicted class for noisy samples in ImageNet-64 when training with the noise detection algorithm proposed in [136]

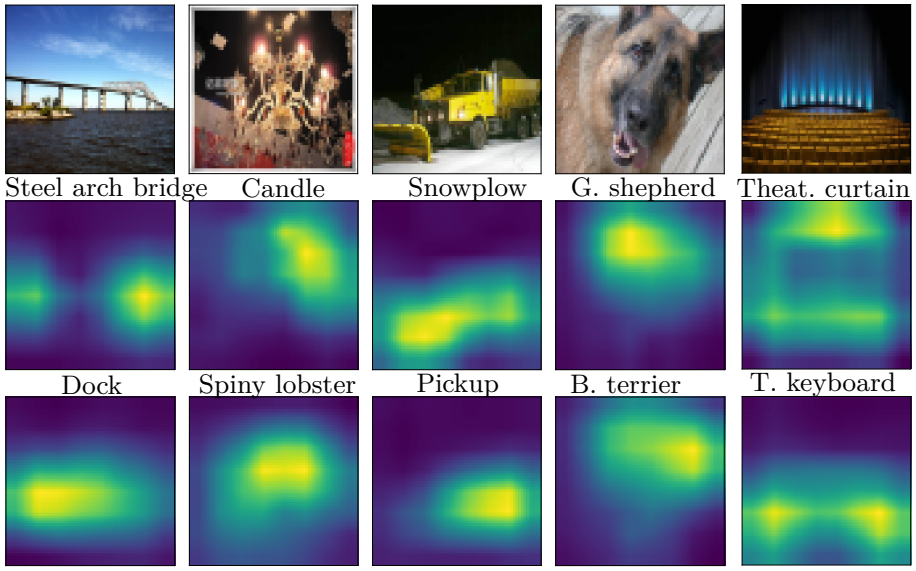


Figure 5.9: Label noise memorization for undetected noisy images in ImageNet-64 (50% asymmetric noise). From top to bottom: image and Class Activation Map (CAM) for the true and predicted class. Note that the predicted class (bottom) is also the noisy label used during training. Image obtained from [136].

(i.e. sample relabeling based on [175]), which illustrate the attention distribution of the CNN across the visual features of each image for the corresponding class. The CAM obtained in Figure 5.9 shows the regions of the images that produce a highest activation value for the corresponding label, i.e. the maps in the second row show the regions of the image that most activate the correct label and the maps in the third row, the regions that most activate the noisy label. This visualization suggests that in the presence of noise, the network still learns patterns in the image that justify memorizing a sample: in particular, the third row shows how the network pays more attention to visual features that confirm the corresponding noisy label used during training and ignores features that would incline the decision towards the true label (provided in the second row). The observations in Figure 5.9 complement previous findings regarding the ability of CNNs to overfit symmetric noise [216] and illustrate the mechanisms by which CNNs overfit to noisy samples when these share structure with the samples that truly belong to the given label; i.e. when trained under asymmetric noise. These experiments correspond to 50% asymmetric noise and are further analyzed in [136].

Finally, a secondary finding from [136, 137] that is worth remarking on suggests

that regularization techniques might be an essential ingredient in designing label noise robust models: experiments comparing state-of-the-art approaches in real-world datasets show that methods that include stronger regularization techniques are able to better leverage the visual features of images and avoid overfitting to the noisy labels. As reported in Table 5.10, *mixup* alone is among the top-performing approaches without any specific label noise technique, followed by [136], which includes a pseudo-labeling stage combined with *mixup*; [103], which also includes *mixup*; and [137], which includes regularization in the form of interpolation training (*mixup*) on top of contrastive learning with strong data augmentation policies.

5.4 Conclusion

The tendency of recent approaches [103, 137] to combine different training strategies suggests that to successfully learn robust representations in the presence of label noise, there needs to be a complex system that accounts for several factors. First, early stages of CNN training are more robust to label noise corruptions, which, as demonstrated in this chapter, could be leveraged with bootstrapping techniques. Other approaches, however, opt for two-stage alternatives [136, 137]: label noise detection followed by semi-supervised learning. This introduces a second important factor for label noise approaches, label noise detection. While several methods address the problem without a noise detection stage [141, 175, 213], adding this stage to the training has the potential of providing stronger guidance for the clean samples and reducing the harmful influence of the noisy ones. The third major factor involved in label noise approaches is regularization, as shown in several studies [218, 10, 103, 110, 137], it provides a significant boost in accuracy when training under label noise.

As shown in this chapter, the dependence of the label corruptions with the images deeply affects the training of CNN. Hence, a crucial step for this field to keep evolving is to move to more realistic noise distributions. While symmetric and asymmetric noise help understanding certain behaviors of CNN in the presence of label noise, they

lack two main factors: instance-dependence noise, and out-of-distribution samples. These are two key elements for future research in this field as they are unavoidable challenges when automating the labeling process of large datasets. Potential initial points are research on more complex dependencies between labels and images, e.g. instance-dependent noise distributions, [191, 203, 35, 192] and out-of-distribution samples exploration [194, 179, 154, 212]. Chapter 7 provides comments on these directions.

5.5 Summary

The focus of this chapter is to explore the training of CNNs in the presence of label noise through the study of the approach introduced in [10]. In particular, we leverage mixture models (concretely a beta mixture model) as an unsupervised way of modeling the two components of the distribution of loss values: one component with the higher loss values and another with the lower, corresponding to noisy and clean samples respectively. A mixture model provides an estimation of the probability of a sample belonging to one or another component, i.e. the probability of a sample being clean or noisy, which we use to dynamically guide a bootstrapping strategy to mitigate the effect of noisy samples in the training: each label becomes the linear weighted combination of the current label and the network prediction. This approach is integrated with *mixup* data augmentation, which results in a very robust approach to label noise.

Additionally, this chapter provides insights on the training of CNN under label noise, including the findings reported in [103, 137] regarding noise distributions and label noise detection. While the main experiments demonstrate the generalization of the proposed approach in [10] on CIFAR, mini-ImageNet, and Clothing1M, further experiments use WebVision, mini-ImageNet-cwln, and ImageNet-32/64 to explore more realistic noise distributions and motivate future research in this direction.

Chapter 6

Budgeted training

The availability of vast amounts of labeled data is crucial when training deep neural networks (DNNs) [117, 206]. Despite prompting considerable advances in many computer vision tasks [211, 169], this dependence poses two challenges: the generation of the datasets and the large computation requirements that arise as a result. Research addressing the former has experienced great progress in recent years via novel techniques that reduce the strong supervision required to achieve top results [174, 184] by improving self-supervised learning, semi-supervised learning, or training with noisy web labels as discussed in Chapters 3, 4, and 5. The latter challenge has also experienced many advances from the side of network efficiency via DNN compression [40, 107], neural architecture search [174, 26], or parameter quantization [148, 82]. All these approaches are designed with a common constraint: a large dataset is needed to achieve top results [206]. This conditions the success of the training process on the available computational resources. Conversely, a smart reduction of the number of samples used during training can alleviate this constraint [91, 122].

The selection of samples plays an important role in the optimization of CNN parameters during training, where SGD is often used. SGD guides the parameter updates using the estimation of model error gradients over sets of samples (mini-batches) that are uniformly randomly selected in an iterative fashion (see Chapter 2 for an in depth explanation of SGD). This strategy assumes equal importance

across samples, whereas other works suggest that alternative strategies for revisiting samples are more effective in achieving better performance [28, 92] and faster convergence [91, 84]. Similarly, the selection of a unique and informative subset of samples (core-set) [183, 37] can reduce the computation requirements during training, while reducing the performance drop with respect to training on all data. However, although removing data samples speeds up training, precise sample selection often requires a pretraining stage that acts counter computational reduction [122, 157].

A possible solution to this limitation might be to dynamically change the important subset during training, as is done by importance sampling methods [7, 219], which select the samples based on a sampling distribution that evolves with the model and often depends on the loss value or network predictions [111, 87]. An up-to-date sample importance estimation is key for current methods to succeed but, in practice, is infeasible to compute [91]. The importance of a sample changes after each iteration and estimations become out-dated, yielding considerable performance drops [28, 219]. Importance sampling methods focus on training with the most relevant samples and achieve a convergence speed-up as a side effect. They do not, however, strictly study the benefits on DNN training when restricting the number of training iterations, i.e. the budget.

Budgeted training [127, 88, 104] imposes an additional constraint on the optimization of a DNN: a maximum number of iterations. Defining this budget provides a concise notion of the limited training resources. Li *et al.* [104] propose to address the budget limitation using specific learning rate schedules that better suit this scenario. Despite the standardized scenario that budgeted training poses to evaluate methods when reducing the computation requirements, there are few works to date in this direction [104, 91]. As mentioned, importance sampling methods are closely related, but the lack of exploration of different budget restrictions makes these approaches less applicable: the sensitivity to hyperparamenters that they often exhibit limits their generalization [28, 111].

This chapter addresses the limitations outlined above by analyzing the effective-

ness of importance sampling methods when a budget restriction is imposed [104]. Given a budget restriction, we study synergies between importance sampling and data augmentation [172, 38, 218]. We find the improvements of importance sampling approaches over uniform random sampling are not always consistent across budgets and datasets. We argue and experimentally confirm (see Section 6.2.5) that when using certain data augmentation strategies [172, 38, 218], existing importance sampling techniques do not provide further benefits, making data augmentation the most effective strategy to exploit a given budget.

The research presented in this Chapter is an adaptation of work published in the British Machine Vision Conference (BMVC) 2021 [12] and the code to replicate the experiments is available at `https://git.io/JKHa3`. Section 6.1 describes the budgeted training paradigm and the adaptations used to evaluate importance sampling techniques under budget restrictions. Section 6.2 empirically compares the efficiency of alternatives to the vanilla SGD with different data augmentation techniques when training under different budget restrictions. Finally, Section 6.3 provides a discussion on the findings of this chapter and on possible future research to better exploit computational resources.

6.1 Training under budget constraints

The experiments presented in this chapter follow the practices described across the thesis: models are trained by gradient based minimization of cross-entropy

$$\mathcal{L}(W) = -\frac{1}{n}\sum_{i=1}^{n} \mathbf{y}_i^T \log h_W(\mathbf{y}|\mathbf{x}_i), \tag{6.1}$$

where n is the number of samples in the dataset $D = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{n}$, where $\mathbf{y}_i \in \{0,1\}^c$ is the one-hot encoding ground-truth label for sample $\mathbf{x}_i$, and c is the number of classes. In this chapter we consider the output $h_W(\mathbf{y}|x_i)$ as the predicted posterior probability of a CNN model given $\mathbf{x}_i$ (i.e. the prediction after *softmax* normalization as in previous chapters), and W are the parameters of the model. While in previous

chapters convergence to a reasonable level of performance determines the end of the training, in budgeted training there is a fixed iteration budget. In this chapter, we adopt the setting defined by [104], where the budget is defined as a percentage of the full training setup. Formally, we define the budget $b \in [0, 1]$ as the fraction of forward and backward passes used for training the model $h_W(\mathbf{x})$ with respect to a standard full training. As we aim at analyzing importance sampling, the budget restriction will be mainly applied to the amount of data $n \times b$ seen every epoch. However, a reduction on the number of epochs t to $t \times b$ (where an epoch t is considered a pass over all samples) is also considered as truncated training for budgeted training.

Truncated training is the simplest approach to budgeted training: keep the standard SGD optimization and reduce the number of epochs trained by the model to $t \times b$. In this chapter we refer to this strategy, where the model sees all the samples every epoch, as *scan-SGD*. While seeing all the samples is common practice, we remove this constraint and draw the samples from a uniform probability distribution at every epoch and call this strategy *unif-SGD*. In this approach the budget is defined by randomly selecting $n \times b$ samples every epoch (and still training for t epochs). As a result *scan-SGD* and *unif-SGD* always train the same number of iterations defined by the budget: the former has longer epochs and restricts the number of them and the latter reduces the number of iterations per epoch.

Importance sampling aims to accelerate the convergence of SGD by sampling the most difficult samples $D_s = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{n_s}$ more often, where $n_s = n \times b$ (the number of samples selected given a certain budget). Loshchilov and Hutter [111] proposed a simple approach for importance sampling that uses the loss of every sample as a measure of the sample importance. Chang *et al.* [28] adapts this approach to avoid additional forward passes by using as importance:

$$p_i^t = \frac{1}{t} \sum_{k=1}^{t} \left(1 - \mathbf{y}_i^T h_W^k(\mathbf{y}|\mathbf{x}_i)\right) + \epsilon^t, \tag{6.2}$$

where $h_W^k(\mathbf{y}|\mathbf{x}_i)$ is the prediction of the model given the sample $\mathbf{x}_i$ in epoch k, and t is the current epoch. Therefore, the average predicted probability across previous

epochs associated to the ground-truth class of each sample defines the importance of sample $\mathbf{x}_i$. The smoothing constant ϵ^t is defined as the mean per-sample importance up to the current epoch: $\frac{1}{n}\sum_{i=1}^{n} p_i^t$. The sampling distribution P^t at a particular epoch t is then given by:

$$P_i^t = \frac{p_i^t}{\sum_{j=1}^{n} p_j}. \tag{6.3}$$

By drawing samples from the distribution P^t, this approach biases the training towards the most difficult samples, and selects samples with highest loss value; we name this method *p-SGD*. Similarly, Chang *et al.* [28] propose to select those samples that are closer to the decision boundaries and favor samples with higher uncertainty by defining the importance measure as $c_i^t = p_i^t \times (1 - p_i^t)$; we name this approach *c-SGD*.

Both *p-SGD* and *c-SGD* are very computationally efficient as the importance estimation only requires information available during training. Conversely, Jiang *et al.* [84] propose to perform forward passes on all the samples to determine the most important ones and later reduce the number of backward passes; they name this method selective backpropagation (*SB*). At every forward pass, *SB* stores the sample $\mathbf{x}_i$ with probability:

$$s_i^t = \left[F_r(\mathcal{L}(h_W^t(\mathbf{x}_i), \mathbf{y}_i))\right]^\gamma, \tag{6.4}$$

where F_r is the cumulative distribution function from a history of the loss values of the last r samples seen by the model and $\gamma > 0$ is a constant that determines the selectivity of the method, i.e. the budget used during the training. In practice, *SB* does as many forward passes as needed until it has enough samples to form a full a mini-batch. It then performs the training forward and backward passes with the selected samples to update the model.

Finally, as an alternative training paradigm to prioritize the most important samples, Kawaguchi and Lu [92] propose to use only the q samples with highest loss from a mini-batch in the backward pass. As the training accuracy increases, q decreases until only 1/16 of the images in the mini-batch are used in the backward

pass. The authors name this approach *ordered* SGD (*OSGD*) and provide a default setting for the adaptive values of q, which change depending on the training accuracy.

Importance sampling methods under budgeted training give a precise notion of the training budget. For *unif-SGD*, *p-SGD*, and *c-SGD* the adaptation needed consists of selecting a fixed number of samples per epoch $n \times b$ based on the corresponding sampling probability distribution P_t and still train the full t epochs. For *SB*, the parameter γ determines the selectivity of the algorithm: higher values will reject more samples. Note that this method requires additional forward passes that we exclude from the budget as they do not induce the backward passes used for training. By considering that each backward pass is twice as computationally expensive as a forward pass we could approximate the budget used by *SB* as $b_{SB} = b + 1/3$, e.g. the results under $b = 0.2$ for *SB* correspond to $b \approx 0.5$ for the other approaches. We adapt *OSGD* by truncating the training as in *scan-SGD*: all the parameters are kept constant but the total number of epochs is reduced to $t \times b$. We also consider the wall-clock time with respect to a full budget training as a metric to evaluate the approaches.

6.2 Experiments and Results

6.2.1 Experimental framework

Datasets

We experiment on image classification tasks using CIFAR-10/100 [98], SVHN [129], and mini-ImageNet [190] datasets. Since the budget training setup does not require any alteration in the dataset, such as discarding or swapping labels as done in previous chapters, the only pre-processing consist of the standardization of the images and the widely used data augmentation techniques that we regard as *standard data augmentation*: random cropping with padding of four pixels per side and random horizontal flip (except in SVHN, where horizontal flip is omitted).

Training details

We train a ResNet-18 architecture [72] for 200 epochs with SGD with momentum of 0.9 and a batch size of 128. We use two learning rate schedules: step-wise and linear decay. For both schedules we adopt the budget-aware version proposed by Li *et al.* [104] and use an initial learning rate of 0.1. In the step-wise case, the learning rate is divided by 10 at 1/3 (epoch 66) and 2/3 (epoch 133) of the training. The linear schedule decreases the learning rate value at every iteration linearly from the initial value to approximately zero (10^{-6}) at the end of the training. We always report the average accuracy and standard deviation of the model across three independent runs. For each budget, we report best results in bold and best results in each section – data augmentation or learning rate schedule – in blue (baseline SGD is excluded).

6.2.2 Budget-free training for importance sampling

Current importance sampling methods from the state-of-the-art are optimized with no restriction in the number of training iterations. While this allows the methods to better exploit the training process, it makes it difficult to evaluate their computational benefit. Therefore, Table 6.1 presents the performance, wall-clock time, and speed-up relative to a full training of the methods presented in Section 6.1.

All methods train with a step-wise linear learning rate schedule. SGD corresponds to a standard training as described in Subsection 6.2.1. *p-SGD* and *c-SGD* correspond to the methods described in Section 6.1 introduced by Chang *et al.* [28] that for the experiments in Table 6.1 train for 200 epochs where the first 70 epochs consist of a warm-up stage with a uniform sampling strategy as done in the original paper. For CIFAR-10 we use a budget of 0.8 for *p-SGD* and 0.7 for *c-SGD*, and for CIFAR-100 a budget of 0.9 for both approaches (budgets retaining most accuracy were selected). Finally, *SB* and *OSGD* follow the setups described in the corresponding papers, [84] and [92], and run on the official code.

While the simpler approaches to importance sampling, *p-SGD* and *c-SGD*, achieve similar performance to SGD and reduce the computational time up to 29.08% (9.93%)

Table 6.1: Test accuracy (%), time (min) and speed-up (%) with respect SGD under a budget-free training. * denotes that we have used the official code.

	CIFAR-10			CIFAR-100		
Method	Accuracy	Time	Speed up	Accuracy	Time	Speed up
SGD	94.58 ± 0.33	141	0.0	74.56 ± 0.06	141	0.0
p-SGD	94.41 ± 0.19	113	19.9	74.44 ± 0.06	127	9.9
c-SGD	94.17 ± 0.11	100	29.1	74.40 ± 0.06	127	9.9
SB (*)	93.90 ± 0.16	85	39.7	73.39 ± 0.37	119	15.6
OSGD (*)	94.34 ± 0.07	139	0.1	74.22 ± 0.21	141	0.0

in CIFAR-10 (CIFAR-100), *SB* reduces the training time 39.72% (15.60%) in CIFAR-10 (CIFAR-100) with very small drops in accuracy. These experiments support observations from previous importance sampling studies: particular hyperparameter configurations of importance sampling approaches effectively reduce computational requirements at minor accuracy expenses.

6.2.3 Budgeted training for importance sampling

We adapt importance sampling approaches as described in Section 6.1 and configure each method to constrain its computation to the given budget. Table 6.2 shows the analyzed methods' performance under the same budget for a step-wise learning rate (SLR) decay and the linear decay (LLR) proposed by Li *et al.* [104] for budgeted training (described in Section 6.2.1). Surprisingly, this setup shows that most methods achieve very similar performance given a predefined budget, thus not observing faster convergence when using importance sampling. Both *p-SGD* and *c-SGD* provide marginal or no improvements: *p-SGD* marginally improves *unif-SGD* in CIFAR-10, but fails to do so in CIFAR-100. Similar behavior is observed in *c-SGD*. Conversely, *SB* surpasses the other approaches consistently for SLR and in most cases in the LLR setup. However, *SB* introduces additional forward passes not considered as budget, while the other methods do not (see Section 6.1 for an estimation of the budget used by *SB*).

We consider *scan-SGD* and *unif-SGD*, as two naive baselines for budgeted training. Despite having similar results (*scan-SGD* seems to be marginally better than *unif-SGD*), we use *unif-SGD* for further experimentation in the remainder of the chapter

Table 6.2: Test accuracy with a step-wise and a linear learning rate decay under different budgets. Note that *SB* requires additional computation (forward passes).

	CIFAR-10			CIFAR-100		
SGD - SLR	94.58 ± 0.33			74.56 ± 0.06		
SGD - LLR	94.80 ± 0.08			75.44 ± 0.16		
Budget:	0.2	0.3	0.5	0.2	0.3	0.5
Step-wise decay of the learning rate (SLR)						
scan-SGD	92.03 ± 0.24	93.06 ± 0.15	93.80 ± 0.15	70.89 ± 0.23	72.31 ± 0.22	73.49 ± 0.20
unif-SGD	91.82 ± 0.05	92.69 ± 0.07	93.71 ± 0.07	70.36 ± 0.30	72.03 ± 0.47	73.36 ± 0.20
p-SGD	92.28 ± 0.05	92.91 ± 0.18	93.85 ± 0.07	70.24 ± 0.28	72.11 ± 0.39	72.94 ± 0.36
c-SGD	91.70 ± 0.25	92.83 ± 0.30	93.71 ± 0.15	69.86 ± 0.36	71.56 ± 0.27	73.02 ± 0.34
SB	93.37 ± 0.11	93.86 ± 0.27	94.21 ± 0.13	70.94 ± 0.38	72.25 ± 0.68	73.39 ± 0.37
OSGD	90.61 ± 0.31	91.78 ± 0.30	93.45 ± 0.10	70.09 ± 0.25	72.18 ± 0.35	73.39 ± 0.22
Linear decay of the learning rate (LLR)						
scan-SGD	92.95 ± 0.07	93.55 ± 0.21	94.22 ± 0.16	**72.04 ± 0.42**	72.97 ± 0.07	73.90 ± 0.43
unif-SGD	92.83 ± 0.14	93.48 ± 0.05	93.98 ± 0.11	72.02 ± 0.24	72.74 ± 0.57	73.93 ± 0.16
p-SGD	93.23 ± 0.14	93.63 ± 0.04	94.14 ± 0.11	71.72 ± 0.37	72.94 ± 0.37	74.06 ± 0.10
c-SGD	92.95 ± 0.17	93.54 ± 0.07	94.11 ± 0.24	71.37 ± 0.49	72.33 ± 0.18	73.93 ± 0.35
SB	**93.78 ± 0.11**	**94.06 ± 0.37**	**94.57 ± 0.18**	71.96 ± 0.67	**73.11 ± 0.42**	**74.35 ± 0.34**
OSGD	91.87 ± 0.36	93.00 ± 0.08	93.93 ± 0.22	71.25 ± 0.11	72.56 ± 0.36	73.40 ± 0.14

as it adopts a uniform random sampling distribution, which allows for a better comparison with the importance sampling methods. Additionally, Table 6.2 confirms the effectiveness of a linear learning rate schedule as proposed in [104]: all methods consistently improve with this schedule and, in most cases, *unif-SGD* and LLR perform on par with *SB* and SLR, and surpasses all the other methods when using SLR.

6.2.4 Train-test bias in importance sampling

The failure of the sampling strategies to consistently outperform *unif-SGD* could be explained by importance sampling breaking the assumption that samples are i.i.d: SGD assumes that a set of randomly selected samples represents the whole dataset and provides an unbiased estimation of the gradients. Importance sampling explicitly breaks this assumption and biases the gradient estimates. While this might produce gradient estimates that have a bigger impact on the loss, breaking the i.i.d. assumption leads SGD to biased solutions [111, 28, 219], which offsets the possible benefits of training with the most relevant samples. As a result, importance sampling does not bring a consistent speed-up in training. Note that approaches that weight

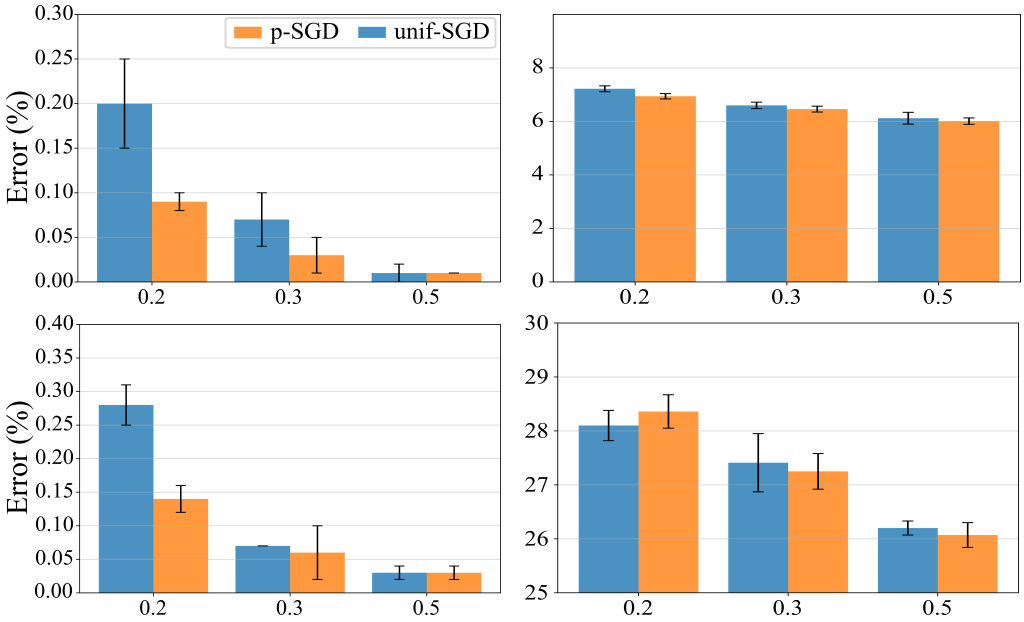


Figure 6.1: Error comparison for *unif-SGD* and *p-SGD* training under different budget restrictions: error in the training (left) and testing (right) set for CIFAR-10 (up) and CIFAR-100 (bottom). Results run over three random seeds, we report average error and standard deviation.

the contribution of each sample with the inverse sampling probability to generate an unbiased gradient estimate obtain similar results [3, 56, 28, 87, 219].

To further explore this idea, Figure 6.1 reports training and testing accuracy of *unif-SGD* and *p-SGD* for CIFAR-10 and CIFAR-100. These results suggest that in some cases importance sampling approaches (e.g. *p-SGD*) provide a substantial benefit in training accuracy that does not translate to the unseen samples in the test set. This supports the claim that breaking the i.i.d. assumption might result in models trained on biased subsets of the data: these models fit the biased statistics of the training set and fail to generalize to the testing set.

6.2.5 Data variability importance during training

Core-set selection approaches [183, 37] aim to find the most representative samples in the dataset to make training more efficient, while keeping accuracy as high as possible. Figure 6.2 presents how core-set selection (Core-set) and a randomly chosen subset (Random) both under-perform uniform random sampling of a subset each epoch (*unif-SGD*), which approaches standard training performance (black dashed

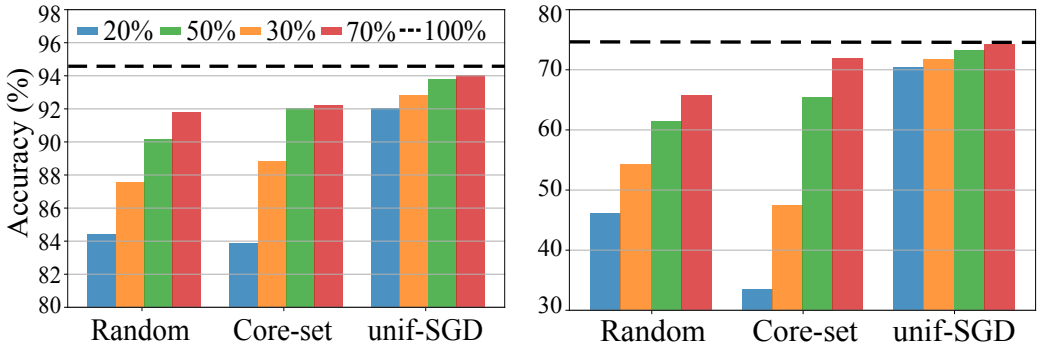


Figure 6.2: Importance of data variability in CIFAR-10 (left) and CIFAR-100 (right) and (d). Comparison of different training set selection strategies: randomly selecting samples at every epoch (*unif-SGD*) outperforms fixed core-set or random subsets

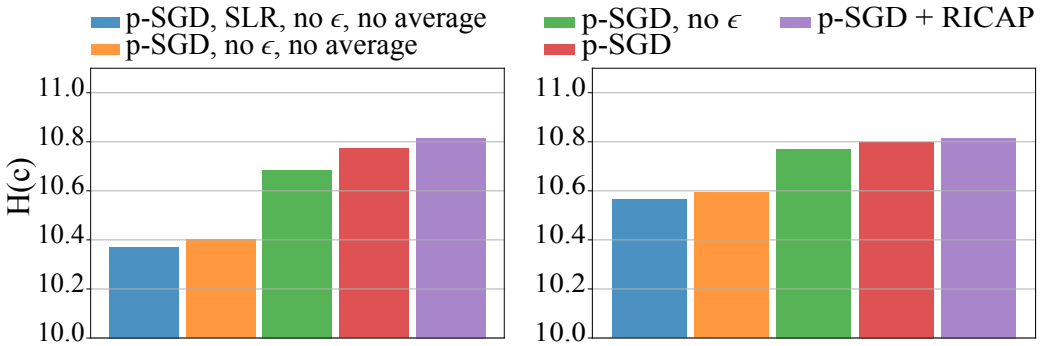


Figure 6.3: Importance of data variability in CIFAR-10 (left) and CIFAR-100 (right). Comparison of the data variability of different training strategies: the entropy of sample counts during training (0.3 budget) demonstrates that importance sampling, linear learning rate, and data augmentation contribute to higher data variability (entropy).

line). This shows that randomly selecting a different subset every epoch (unif-SGD), which is equally computationally efficient, achieves substantially better accuracy. This result supports the widely adopted assumption that data variability is key and suggests that it might be more important than sample quality.

We also find data variability to play an important role within importance sampling. Figure 6.3 shows data variability measured using the entropy $H(c)$ of the number of times that a sample is seen by the network during training, with c being the N-D distribution of sample counts. These results show how increases in variability (higher entropy) follow accuracy improvements in *p-SGD* when introducing the LLR, the smoothing constant to the P^t sampling distribution, the average of the predictions across epochs, and data augmentation.

6.2.6 Data augmentation for importance sampling

In this section, we explore the role of data augmentation techniques in budgeted training scenarios. Given the importance of data variability observed in Subsection 6.2.5, this is a crucial step to understand how to better exploit limited computational resources for training CNN. In particular, this section shows how data augmentation present a better alternative than importance sampling techniques to leverage limited computational resources. Importance sampling approaches usually do not explore the interaction of sampling strategies with data augmentation techniques [111, 91, 84]. To better understand this interaction, we explore interpolation-based augmentations via RICAP [172] and *mixup* [218], where samples and the corresponding labels are linearly combined to smooth the decision boundaries between classes; and non-interpolation augmentations using RandAugment [38], which provides a considerably strong augmentation by randomly combining several augmentations from a pool of the most widely used data augmentation techniques in the literature. We implemented these data augmentation policies as reported in the original papers (see Table 6.3 for the hyperparameters used in our experiments). Note that for *mixup* and RICAP we combine two and four images respectively within each mini-batch, which results in the same number of samples being shown to the network ($t \times b$).

Tables 6.3 and 6.4 show that data augmentation is beneficial in a budgeted training scenario: in most cases all strategies increase performance compared to standard data augmentation. The main exception is for the lowest budget for *SB* where in some cases data augmentation hurts performance. In particular, with RICAP and *mixup*, the improvements from importance sampling approaches are marginal and the naive *unif-SGD* provides results close to full training with standard augmentation. In some cases, *unif-SGD* surpasses full-training with standard augmentations, e.g. RICAP with 0.3 and 0.5 budget and both *mixup* and RICAP with 0.3 budget in CIFAR-10/100. This is even more evident in SVHN where all the budgets in Table 6.4 for *unif-SGD* with RICAP surpass full training (SGD) with standard augmentation.

Given that the cost of the data augmentation policies used is negligible (see

Table 6.3: Data augmentation for budgeted importance sampling in CIFAR-10 and CIFAR-100. N and M are the number and strength of RandAugment augmentations, and α controls the interpolation in *mixup* and RICAP. Note that SGD corresponds to the full training.

	CIFAR-10			CIFAR-100		
Budget:	0.2	0.3	0.5	0.2	0.3	0.5
			Standard data augmentation			
SGD ($b = 1$)		94.80 ± 0.08			75.44 ± 0.16	
unif-SGD	92.83 ± 0.14	93.48 ± 0.05	93.98 ± 0.11	72.02 ± 0.24	72.74 ± 0.57	73.93 ± 0.16
p-SGD	93.23 ± 0.14	93.63 ± 0.04	94.14 ± 0.11	71.72 ± 0.37	72.94 ± 0.37	74.06 ± 0.10
SB	93.78 ± 0.11	94.06 ± 0.37	94.57 ± 0.18	71.96 ± 0.67	73.11 ± 0.42	74.35 ± 0.34
			RandAugment data augmentation (N = 2, M = 4)			
SGD ($b = 1$)		95.56 ± 0.12			75.52 ± 0.17	
unif-SGD	92.76 ± 0.16	93.78 ± 0.11	94.64 ± 0.08	71.44 ± 0.37	73.23 ± 0.29	74.78 ± 0.45
p-SGD	92.95 ± 0.31	93.99 ± 0.28	94.91 ± 0.18	71.63 ± 0.27	72.91 ± 0.13	74.30 ± 0.04
SB	93.27 ± 0.38	94.64 ± 0.07	95.27 ± 0.26	66.84 ± 1.15	73.79 ± 0.40	74.87 ± 0.18
			mixup data augmentation ($\alpha = 0.3$)			
SGD ($b = 1$)		95.82 ± 0.17			77.62 ± 0.40	
unif-SGD	93.64 ± 0.27	94.49 ± 0.04	95.18 ± 0.05	73.28 ± 0.51	75.13 ± 0.52	75.80 ± 0.34
p-SGD	93.78 ± 0.04	94.41 ± 0.16	95.26 ± 0.06	73.35 ± 0.29	75.05 ± 0.15	75.87 ± 0.15
SB	93.62 ± 0.36	93.92 ± 0.08	94.51 ± 0.17	73.38 ± 0.13	74.88 ± 0.31	75.57 ± 0.23
			RICAP data augmentation ($\alpha = 0.3$)			
SGD ($b = 1$)		96.17 ± 0.09			78.91 ± 0.07	
unif-SGD	93.85 ± 0.10	**94.93 ± 0.29**	95.47 ± 0.18	**74.87 ± 0.28**	76.27 ± 0.32	**77.83 ± 0.15**
p-SGD	**94.02 ± 0.18**	94.79 ± 0.18	**95.63 ± 0.15**	74.59 ± 0.15	**76.50 ± 0.22**	77.58 ± 0.49
SB	89.93 ± 0.84	93.64 ± 0.42	94.76 ± 0.02	56.66 ± 0.65	72.24 ± 0.58	76.26 ± 0.22

Table 6.5 for the wall-clock times when $b = 0.3$), our results show that adequate data augmentation alone can reduce training time at no cost in accuracy and in some cases with a considerable increase in accuracy. For example, a 70% reduction in training time (0.3 budget) corresponds to an increase in accuracy from 75.44% to 76.27% in CIFAR-100 and from 94.80% to 94.93% in CIFAR-10. Also, a 50% reduction (0.5 budget) corresponds to an increase in accuracy from 75.44% to 77.83% in CIFAR-100 and from 94.80% to 95.47% in CIFAR-10.

We also experimented with extremely low budgets (see Table 6.6) and found that importance sampling approaches (*p-SGD* and *SB*) still bring little improvement over uniform random sampling (*unif-SGD*). Here, additional data augmentation does not bring a significant improvement in accuracy and in the most challenging cases, hinders convergence. For example, when introducing RICAP with $b = 0.05$, the accuracy drops approximately two points in accuracy in CIFAR-10, five points in CIFAR-100, and seven points in mini-ImageNet with respect 87.90%, 62.66%, and

Table 6.4: Data augmentation for budgeted importance sampling in SVHN and mini-ImageNet. N and M are the number and strength of RandAugment augmentations, and α controls the interpolation in *mixup* and RICAP.

	SVHN			mini-ImageNet		
Budget:	0.2	0.3	0.5	0.2	0.3	0.5
Standard data augmentation						
SGD ($b = 1$)		97.02 ± 0.05			75.19 ± 0.16	
unif-SGD	96.56 ± 0.12	96.78 ± 0.13	96.95 ± 0.07	70.87 ± 0.56	72.19 ± 0.43	73.88 ± 0.42
p-SGD	96.52 ± 0.03	96.75 ± 0.03	96.84 ± 0.06	71.05 ± 0.29	72.39 ± 0.45	73.66 ± 0.39
SB	96.93 ± 0.07	96.85 ± 0.01	96.97 ± 0.06	69.68 ± 0.09	71.46 ± 0.15	73.51 ± 0.30
RandAugment data augmentation (N = 2, M = 4)						
SGD ($b = 1$)		97.59 ± 0.14			74.15 ± 0.22	
unif-SGD	97.38 ± 0.05	97.50 ± 0.07	97.60 ± 0.05	71.29 ± 0.25	73.04 ± 0.34	73.21 ± 0.52
p-SGD	97.25 ± 0.03	97.44 ± 0.02	97.52 ± 0.03	71.43 ± 0.25	72.36 ± 0.15	73.21 ± 0.38
SB	97.42 ± 0.09	97.43 ± 0.19	97.56 ± 0.05	67.17 ± 2.51	71.69 ± 0.31	73.28 ± 0.03
mixup data augmentation ($\alpha = 0.3$)						
SGD ($b = 1$)		97.24 ± 0.03			76.28 ± 0.28	
unif-SGD	96.99 ± 0.09	97.04 ± 0.08	97.24 ± 0.07	72.50 ± 0.51	73.76 ± 0.26	75.05 ± 0.29
p-SGD	96.92 ± 0.08	97.34 ± 0.49	97.37 ± 0.49	72.21 ± 0.81	73.63 ± 0.13	74.54 ± 0.53
SB	96.80 ± 0.09	96.92 ± 0.09	96.96 ± 0.09	70.12 ± 0.51	72.01 ± 0.72	73.76 ± 0.36
RICAP data augmentation ($\alpha = 0.3$)						
SGD ($b = 1$)		97.61 ± 0.06			78.75 ± 0.40	
unif-SGD	97.47 ± 0.04	**97.62 ± 0.16**	97.55 ± 0.04	73.56 ± 0.24	75.15 ± 0.45	77.20 ± 0.33
p-SGD	**97.48 ± 0.08**	97.45 ± 0.06	**97.57 ± 0.05**	**73.67 ± 0.60**	**75.46 ± 0.27**	**77.25 ± 0.47**
SB	97.34 ± 0.03	97.40 ± 0.06	97.45 ± 0.01	53.26 ± 0.71	71.75 ± 0.67	75.65 ± 0.40

Table 6.5: Wall-clock time (minutes) in CIFAR-100 for a training of 0.3 of budget. Note that SGD corresponds to a baseline that uses the full budget and standard data augmentation.

Approaches:	*unif-SGD*	*p-SGD*	*SB*
SGD ($b = 1$)		141	
Standard data augmentation	47	48	91
RandAugment [38]	48	48	93
mixup [218]	48	48	93
RICAP [172]	49	49	95

56.38% for *unif-SGD* with standard data augmentation.

6.3 Conclusion

The research described in this chapter exposes the complexity of efficiently training CNNs. While some importance sampling techniques seem to provide substantial improvements for CNN optimization and reduce computational costs, these are very dependent on training configurations and datasets. In particular, when adapted to budget restricted setups, importance sampling fails to consistently accelerate

Table 6.6: Test accuracy for CIFAR-10/100 and mini-ImageNet under extreme budgets.

	CIFAR-10		CIFAR-100		mini-ImageNet	
Budget:	0.05	0.1	0.05	0.1	0.05	0.1
Standard data augmentation						
unif-SGD	87.90 ± 0.40	91.46 ± 0.08	**62.66 ± 0.65**	69.34 ± 0.68	56.38 ± 0.11	67.61 ± 0.52
p-SGD	**88.86 ± 0.17**	91.66 ± 0.11	62.20 ± 0.56	69.32 ± 0.17	**56.95 ± 0.43**	67.67 ± 0.41
SB	79.45 ± 4.31	**92.66 ± 0.14**	50.53 ± 2.27	68.29 ± 0.68	11.19 ± 3.46	61.25 ± 1.76
RandAugment data augmentation (N = 2, M = 4)						
unif-SGD	83.24 ± 0.06	88.95 ± 0.22	47.64 ± 3.34	64.48 ± 0.10	42.35 ± 1.54	64.98 ± 0.47
p-SGD	83.94 ± 0.26	89.77 ± 0.38	48.78 ± 1.48	65.05 ± 0.37	41.72 ± 0.77	65.88 ± 0.15
SB	32.21 ± 4.14	33.86 ± 5.02	5.05 ± 0.64	5.05 ± 0.64	5.61 ± 0.66	5.94 ± 0.13
mixup data augmentation ($\alpha = 0.3$)						
unif-SGD	87.33 ± 0.42	91.74 ± 0.04	59.90 ± 0.71	**70.43 ± 0.45**	53.13 ± 0.83	**68.54 ± 0.98**
p-SGD	87.56 ± 0.67	91.59 ± 0.17	59.68 ± 0.71	70.31 ± 0.10	54.20 ± 0.95	68.39 ± 0.46
SB	77.72 ± 5.31	92.56 ± 0.15	43.27 ± 7.37	69.64 ± 0.24	12.10 ± 0.27	61.01 ± 0.64
RICAP data augmentation ($\alpha = 0.3$)						
unif-SGD	85.61 ± 0.24	91.32 ± 0.28	55.85 ± 0.51	69.43 ± 0.33	48.95 ± 0.65	67.26 ± 0.63
p-SGD	85.57 ± 0.70	90.94 ± 0.16	56.09 ± 0.71	70.05 ± 0.07	49.35 ± 0.60	67.27 ± 0.85
SB	44.93 ± 2.67	54.76 ± 4.31	10.75 ± 0.72	13.33 ± 0.39	8.71 ± 0.45	10.84 ± 0.86

CNN convergence, especially when considering the additional costs of computing the importance of the samples. We argue that the disagreement between theoretical and experimental results steams from the bias introduced in the dataset when prioritizing a subset of the samples, i.e. those that are more "important" at every step of the training. While this might be beneficial under particular training configurations, it fails across setups and datasets, leading to the primary contribution of this chapter: data variability trumps data importance.

We explore the effect of data variability when the training is limited to a predefined budget of iterations and conclude that prioritizing variety in the samples shown to the model is a more effective option than exploiting the most relevant or important samples during training. In other words, given a limited budget, CNNs benefit the most from data augmentation rather than sample selection approaches. For instance, we show that adequate data augmentation surpasses state-of-the-art importance sampling methods and allows for up to a 70% reduction of the training time (budget) with no loss (and sometimes an increase) in accuracy. This, however, might not apply to cases of extreme budget restrictions where data augmentation might introduce too much difficulty for the CNN to learn useful features. We leave for future work

the exploration of extreme budgets, and the synergies between such cases and other factors that influence convergence, such as learning rate schedules, sample selection policies, or data augmentation techniques.

Finally, we find of particular interest the high accuracy reached under extreme budgets with the standard training (standard data augmentation and *unif-SGD*). For instance, it is remarkable that 10% of the training iterations ($b = 0.1$) in CIFAR-10 (91.46) leads to over 95% of the accuracy reached with 100% of the budget (94.80). These results might be very valuable for more computationally expensive tasks: e.g. classification in ImageNet or video classification. These scenarios often present larger samples that contain more information and ease the representation learning stage of the training: the CNN is able to fit more filter multiplication for each convolution, thus the filters see more variety within an image. This additional variety, however, does not span across samples as additional data augmentation does, but results in larger disparity in the difficulty, or importance, of each sample. This suggest that additional data augmentation in these scenarios might be a good approach to boost performance in budgeted scenarios. Chapter 7 discusses possible future work to extend on these observations and to explore their implications into the optimization process of CNN.

6.4 Summary

This chapter empirically addresses the study of budgeted training as a paradigm to explore the efficiency of training CNNs. Concretely, we explore the role of importance sampling and data augmentation techniques. In Subsection 6.2.2 and 6.2.3, we show a strong dependence of current importance sampling approaches to the training setup. In Subsection 6.2.5 then, we report the need for variety in the data, and in Subsection 6.2.6 the effectiveness of data augmentation techniques when aiming at leveraging a given budget of training iterations to learn the best possible CNN features. Additionally, in this chapter we identify potential future work on exploring extreme budget restrictions and possible synergies with other elements of the training.

Chapter 7

Conclusions

Convolutional neural networks have reached impressive results in many machine learning applications, populated the research landscape in computer vision, and achieved new state-of-the-art performance in most computer vision tasks. These results, however, come with certain requirements: To reach top results convolutional neural networks need large datasets, strong supervision in the form of precise annotations, and long training processes. While it is relatively easy to obtain large amounts of data, the annotation process often requires considerable human intervention and is likely to result in imperfect datasets where labels might be incorrect. Adding to this challenge (and assuming the datasets are properly annotated), convolutional neural networks need to visit every sample several times, which results in training processes that could last from a few hours to several days depending on the size of the dataset and the CNN model. The research presented in this thesis addresses these two challenges: Chapters 3, 4, and 5 address alternatives to reduce the dependence to strong supervision during training, and Chapter 6 address alternatives to reach top accuracy while reducing the computation requirements.

Concretely, in Chapter 3, we provide a study of self-supervised learning, i.e. where convolutional neural networks train directly from the images and the supervision comes from visual properties in the images. In particular, the approach studied, Exemplar-CNN [47], replaces the label supervision with surrogate classes: it encourages the CNN to predict transformations of patches from an image as belonging to

a particular instance: the original sample. In this chapter, we propose a clustering algorithm to group similar images and to allow the Exemplar-CNN to leverage larger amounts of unlabeled data. In Chapter 4 we eased the supervision restriction and assumed the availability of a small labeled set of samples, i.e. semi-supervised learning. In this case, the proposed approach addresses confirmation bias as the main drawback of pseudo-labeling techniques – approaches that leverage model predictions to further guide the training. This chapter shows that pseudo-labeling techniques can reach state-of-the-art results when properly regularized. The focus of Chapter 5 is the training of convolutional neural networks under the presence of label noise: all samples are labeled but potentially incorrectly. In this chapter, we present a bootstrapping method and a noise modeling approach to identify the likelihood of a sample being noisy, i.e. incorrectly labeled. In this chapter we also provide insights into the label noise setup and the interaction between different noise distributions and CNN training. Finally, in Chapter 6 we assumed correct labels in the training set and a restricted number of iterations to train the model, i.e. budgeted training. Particularly, this chapter focuses on techniques that prioritize the most important samples to better leverage the given budget and the corresponding experiments reveal that in these conditions data diversity is a crucial factor.

In this chapter, Section 7.1 address the hypothesis described in Chapter 1, and how the research presented in Chapters 3 to 6 addresses the hypothesis through the research questions also introduced in Chapter 1. Section 7.2 summarizes the research contributions of this thesis. Section 7.3 elaborates on the suggestions for future research introduced in the main chapters of the thesis. Finally, Section 7.4 provides the closing remarks for this thesis.

7.1 Hypothesis and research questions

The hypotheses introduced in Chapter 1 are discussed in this section in light of the research presented in the corresponding chapters. Each of the research questions associated with each hypothesis is addressed to provide a more concise notion of the

contribution of this thesis.

Hypothesis 1

Given a large set of unlabeled images, clustering algorithms aid the self-supervised training of neural networks by enforcing the assumption that images belonging to the same class, or images with similar visual features, are closer than those that are from different classes when mapped to the representation space learned by the neural network.

Research question 1: ***How could the clustering assumption be enforced in the self-supervised training of neural networks? Could neural networks benefit from clustering the features learned during a self-supervised training?***

Experiments described in Chapter 3 show the benefits that clustering algorithms yield in a self-supervised scenario in the STL-10 benchmark [36]. The proposed approach applies a clustering algorithm to self-supervised CNN features to obtain visually similar images. These images are grouped to create a more compact initial step for the self-supervised approach studied, Exemplar-CNN [47], and to allow the introduction of larger amounts of unlabeled data. In this setup, the clustering algorithm enforces the clustering assumption by using the model predictions to cluster together similar images and to reduce the number of initial instances, while including more data in the training. The results reported in Chapter 3 are inline with recent findings [27] that show the benefits of applying clustering techniques while learning representations in a self-supervised fashion. In this scenario, clustering algorithms encourage similar images to be represented by features that are nearby in the representation space.

Hypothesis 2

Neural networks can overcome the dependence on strong supervision from carefully annotated datasets by leveraging a trustworthy small subset of the data and extrapolating

the features learned to the rest of the data.

Research question 2: ***How can neural networks overcome the main limitation when propagating knowledge from a reliable subset of the data to the remaining larger subset? How can one determine which subset is reliable?***

Chapters 4 and 5 empirically show that convolutional neural networks are able to leverage the structure from certain subsets of samples to learn better features: a subset of labeled samples in the former and a potentially clean subset of samples in the latter. Concretely, the experiments described in Chapter 4 show that the availability of a labeled set of samples allows the CNN to leverage vasts amounts of unlabeled data to learn better features. In particular, we conclude that model predictions are a valuable alternative to use as pseudo-labels for the unlabeled samples in the dataset. From the experiments in Chapter 5, we conclude that bootstrapping is also an effective approach to leverage reliable samples in the label noise scenario. This chapter also answers the second part of *Research question 2*, and proposes several approaches to identify reliable subsets of samples.

Hypothesis 3

In scenarios where the training budget is restricted to a certain amount of computation, neural networks do not benefit from seeing the most important samples, but from seeing more variety in the data.

Research question 3: ***Are there some samples more useful than others when training a neural network? Is it possible to leverage them to achieve better performance when the training is limited to a certain budget?***

Experiments conducted in Chapter 6 investigate the effect of prioritizing certain samples during training, as done by importance sampling approaches, when the training is limited to a given budget. From these experiments, we conclude that

certain samples have a higher contribution to the training and that presenting them to the model more frequently accelerates convergence to a better performance. This, however, does not result in an effective speed-up of the training process for two reasons. Firstly, the most important samples change from iteration to iteration and the computation of the sample importance requires additional iterations that offset the gains from selecting the "important" samples. Secondly, prioritizing a subset of samples biases the model and harms generalization.

7.2 Research Contributions and proposed solutions

The contributions of this research are collected and summarized in the following list:

- Chapter 3: self-supervised learning through Exemplar-CNN

    1. We address the main drawback of Exemplar-CNN by reducing the number of near-duplicates in STL-10 through an agglomerative clustering technique that allows the model to scale to larger numbers of unlabeled samples.
    2. Visual exploration of the widely-used unsupervised learning benchmark STL-10 revealed certain drawbacks that should be considered: artifacts in the samples, out-of-distribution samples, and a large number of near-duplicate samples.

- Chapter 4: semi-supervised learning

    1. We identify the main source of feature degradation when training pseudo-labeling approaches: confirmation bias.
    2. We propose an approach to pseudo-label that combines targeted regularizations to reduce the effect of confirmation bias.
    3. Thorough ablation studies of the proposed pseudo-labeling approach provide valuable insights for its application.

- Chapter 5: label noise

    1. We propose a bootstrapping approach to learn robust features in the presence of label noise and a noise detection approach that leverages mixture models to estimate the per sample probability of being noisy.
    2. We provide insights into the training of convolutional neural networks in the presence of different noise distributions and discussed their relevance in real-world noisy datasets.

- Chapter 6: budgeted training

    1. Evaluation of importance sampling approaches under budget constraints shows these approaches to be impractical.
    2. We propose an alternative approach to better leverage a given budget and obtain competitive CNN features.

We also identifiy potential directions for future work in each of the chapters to further push the field towards more realistic scenarios and to improve the applicability of convolutional neural networks in real-world challenges. We elaborate on these observations in Section 7.3.

7.3 Recommendations and future work

We compile in this subsection the main research directions introduced across the thesis, and potential opportunities for future research in these areas:

- Dependence of self-supervised approaches on data augmentation techniques:
    - As demonstrated in [181], image transformations are crucial for contrastive approaches to self-supervised learning – some seem to be more useful than others. This dependence is also observed in Exemplar-CNN, which fails to converge to a reasonable performance if data augmentation is not properly selected. These strong augmentations are very random and vastly increase

the image space making feature learning difficult. Hence, it is natural to think that self-supervised learning approaches would benefit from more specific transformations that ease the training process and allow for less computationally intensive training.

  - In this same direction, since the current data augmentation techniques are specific to images, we find it valuable to develop domain-agnostic approaches that generalize to other types of data, e.g. graphs, or audio. This could include research on input or intermediate feature transformations as well as research on contrastive learning approaches that better leverage these types of transformations.

- Robustness of semi-supervised approaches to distribution mismatch between labeled and unlabeled samples:

  - Current experimental scenarios do not reflect a distribution mismatch between these two subsets, which is very likely when dealing with real-world data: The unlabeled samples are likely to belong to classes that are not in the testing set (out-of-distribution samples), might not belong to the training labeled set (label distribution mismatch), or might not be uniformly distributed across classes (long-tail distributions). Hence, we believe that it is important to further challenge the current semi-supervised methods and evaluate their performance in benchmarks that include these additional challenges. We find particularly interesting the exploration of works on domain adaptation [208, 126, 90] to address distribution mismatch between sets.

- Robustness of label noise approaches to distribution mismatch between clean and noisy samples:

  - Unlike with semi-supervised learning, recent label noise benchmarks already include out-of-distribution samples. However, the methods proposed do not directly tackle the differences in the distributions from these two

subsets. Aside from out-of-distribution samples, a secondary source of mismatch between distributions is the imbalance in the distribution of samples across classes: long-tailed distributions. Addressing this mismatch would result in methods that better generalize to real-world datasets.

  - Current methods could, also, be improved by exploiting the behavior of the network at different stages of the training: some samples might be correctly predicted at early stages and incorrectly after the model fits the noise distribution. While some approaches already consider model predictions across epochs for noise detection, this has not been studied for improving training robustness.

- Further exploration of CNN training under a predefined iteration budget:

  - In this setup, CNN training is amply aided by variety in the data. However, the metrics used in the literature to prioritize certain samples only account for the difficulty, or importance, of those samples and result in models trained on biased subsets of data. Approaches that design mini-batches to account for variation in the data could be beneficial for accelerating the convergence of the training. We believe that this could be an alternative to strong data augmentation policies, which harm convergence when training under extreme budget restrictions.

  - Additionally, the results observed in Chapter 6 when training under extreme budgets, suggest that different data augmentations benefit different budget restrictions and motivate the design of epoch dependent data augmentation techniques: models trained under extreme budgets with weak augmentations could be enhanced by further training stages with stronger augmentation policies.

  - Finally, we find it highly interesting to explore how the conclusions from Chapter 6 scale to larger tasks such as classification in the full size ImageNet dataset or video classification.

7.4 Closing remarks

Recent advances in computer vision have been mostly driven by deep learning methods, including the successful training of convolutional neural networks. These methods have allowed to leverage vast amounts of labeled data to train models able to extract powerful representations from visual data. The research conducted during the development of this thesis focuses the applicability of deep learning methods when the data and training conditions present certain challenges: scarcity or corruption of labels, and computational limitations. The motivation for the former lies in the dilemma of having vasts amounts of available data and the costs of obtaining labeled data to train deep learning models. The latter, however, is motivated by the long training processes required to obtain top performance when labeled data is available. The approaches explored and developed in this thesis are of particular interest for those applications where the training resources are scarce, both in terms of label availability and computational resources. Self-supervised learning, semi-supervised learning and learning with label noise could be of significant benefit to applications where data availability and annotation is most challenging: e.g. medical imaging where due to the need of expert knowledge the annotations are scarce, or self-driving cars where large amounts of data are easily accessible but very costly to annotate. Budgeted training, however, would be a valuable addition to those applications where computation is limited: e.g. research itself, where comparing different computationally expensive algorithms often slows down the exploration of the field, or smart environments where models need to be trained on the edge.

Concretely, this thesis explores methods to train in scenarios where only unlabeled data is available, where a small set of labeled data is available alongside a large set of unlabeled data, where the available labels are unreliable, and where training iterations are restricted to a predefined budget. The author finds particularly interesting the inherent robustness shown by deep learning methods: in the different scenarios, the performance degradation is substantially reduced when the challenges in the dataset are accounted for. For example, when training under label noise, standard methods

show considerable performance degradation, but when the model incorporates label noise techniques, even high levels of label corruption (up to 90%) provide a good training set up for convolutional neural networks. Likewise, when restricting the number of iterations allowed during training (down to 10%), convolutional neural networks with proper data augmentation are able to reach near full performance. By leveraging this robustness, deep learning methods have become a pivotal tool for computer vision and we believe that further exploration of the challenging scenarios mentioned earlier will result in methods that better generalize to practical applications.

Bibliography

[1] I. Sutskever G. Hinton A. Krizhevsky. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2012.

[2] Triantafyllos Afouras, Andrew Owens, Joon Son Chung, and Andrew Zisserman. "Self-supervised learning of audio-visual objects from video". In: *arXiv:2008.04237*. 2020.

[3] Guillaume Alain, Alex Lamb, Chinnadhurai Sankar, Aaron Courville, and Yoshua Bengio. "Variance reduction in sgd by distributed importance sampling". In: *International Conference on Learning Representations (ICLR)*. 2016.

[4] Paul Albert, Diego Ortego, Eric Arazo, Noel E. O'Connor, and Kevin McGuinness. "ReLaB: Reliable Label Bootstrapping for Semi-Supervised Learning". In: *International Joint Conference on Neural Networks (IJCNN)*. 2021.

[5] Görkem Algan and Ilkay Ulusoy. "Image classification with deep learning in the presence of noisy labels: A survey". In: *Knowledge-Based Systems*. Elsevier, 2021.

[6] Mohsan Alvi, Andrew Zisserman, and Christoffer Nellaker. "Turning a blind eye: Explicit removal of biases and variation from deep neural network embeddings". In: *European Conference on Computer Vision Workshops (ECCVw)*. 2018.

[7] Hadi Amiri, Timothy Miller, and Guergana Savova. "Repeat before forgetting: Spaced repetition for efficient and effective training of neural networks". In: *Conference on Empirical Methods in Natural Language Processing*. 2017.

[8] Lasse F. Wolff Anthony, Benjamin Kanding, and Raghavendra Selvan. *Carbontracker: Tracking and Predicting the Carbon Footprint of Training Deep Learning Models*. ICML Workshop on Challenges in Deploying and monitoring Machine Learning Systems. 2020.

[9] E. Arazo, N.E. O'Connor, and K. McGuinness. "Improving unsupervised learning with exemplarCNNs". In: *Irish Machine Vision and Image Processing conference (IMVIP)*. 2019.

[10] E. Arazo, D. Ortego, P. Albert, N. O'Connor, and K. McGuinness. "Unsupervised Label Noise Modeling and Loss Correction". In: *International Conference on Machine Learning (ICML)*. 2019.

[11] E. Arazo, D. Ortego, P. Albert, N.E. O'Connor, and K. McGuinness. "Pseudo-Labeling and Confirmation Bias in Deep Semi-Supervised Learning". In: *International Joint Conference on Neural Networks (IJCNN)*. 2020.

[12] Eric Arazo, Diego Ortego, Paul Albert, Noel E O'Connor, and Kevin McGuinness. "How Important is Importance Sampling for Deep Budgeted training?" In: *British Machine Vision Conference (BMVC)*. 2021.

[13] Eric Arazo, Diego Ortego, Paul Albert, Noel E. O'Connor, and Kevin McGuinness. *How Important is Importance Sampling for Deep Budgeted Training?* `http://openreview.net/forum?id=TqQ0oOzJlai`. 2021.

[14] D. Arpit, S. Jastrzebski, N. Ballas, D. Krueger, E. Bengio, M.S. Kanwal, T. Maharaj, A. Fischer, A. Courville, Y. Bengio, and S. Lacoste-Julien. "A Closer Look at Memorization in Deep Networks". In: *International Conference on Machine Learning (ICML)*. 2017.

[15] Y.M. Asano, C. Rupprecht, and A. Vedaldi. "Surprising Effectiveness of Few-Image Unsupervised Feature Learning". In: *IEEE International Conference on Computer Vision (ICCV)*. 2019.

[16] B. Athiwaratkun, M. Finzi, P. Izmailov, and A.G. Wilson. "There Are Many Consistent Explanations of Unlabeled Data: Why You Should Average". In: *International Conference on Learning Representations (ICLR)*. 2019.

[17] Miguel A Bautista, Artsiom Sanakoyeu, Ekaterina Tikhoncheva, and Bjorn Ommer. "Cliquecnn: Deep unsupervised exemplar learning". In: *Advances in Neural Information Processing Systems*. 2016.

[18] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. "Curriculum learning". In: *International Conference on Machine Learning (ICML)*. 2009.

[19] D. Berthelot, N. Carlini, I. Goodfellow, N. Papernot, A. Oliver, and C. Raffel. "MixMatch: A Holistic Approach to Semi-Supervised Learning". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.

[20] David Berthelot, Nicholas Carlini, Ekin D Cubuk, Alex Kurakin, Kihyuk Sohn, Han Zhang, and Colin Raffel. "Remixmatch: Semi-supervised learning with distribution alignment and augmentation anchoring". In: *arXiv:1911.09785*. 2019.

[21] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[22] Léon Bottou, Frank E Curtis, and Jorge Nocedal. "Optimization methods for large-scale machine learning". In: *Siam Review*. 2018.

[23] John S Bridle, Anthony JR Heading, and David JC MacKay. "Unsupervised classifiers, mutual information and'Phantom Targets'". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 1992.

[24] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. "Signature verification using a" siamese" time delay neural network". In: *Advances in neural information processing systems (NeurIPS)*. 1993.

[25] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. "Language models are few-shot learners". In: *arXiv:2005.14165*. 2020.

[26] Han Cai, Ligeng Zhu, and Song Han. "Proxylessnas: Direct neural architecture search on target task and hardware". In: *International Conference on Learning Representations (ICLR)*. 2019.

[27] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. "Unsupervised learning of visual features by contrasting cluster assignments". In: *Advances in neural information processing systems (NeurIPS)*. 2020.

[28] Haw-Shiuan Chang, Erik Learned-Miller, and Andrew McCallum. "Active bias: Training more accurate neural networks by emphasizing high variance samples". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2017.

[29] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. "A Simple Framework for Contrastive Learning of Visual Representations". In: *International Conference on Machine Learning (ICML)*. 2020.

[30] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton. "Big self-supervised models are strong semi-supervised learners". In: *arXiv:2006.10029*. 2020.

[31] X. Chen, H. Fan, R. Girshick, and K. He. "Improved Baselines with Momentum Contrastive Learning". In: *arXiv:2003.04297*. 2020.

[32] Xinlei Chen and Kaiming He. "Exploring Simple Siamese Representation Learning". In: *arXiv:2011.10566*. 2020.

[33] Y. Chen, X. Zhu, and S. Gong. "Semi-Supervised Deep Learning with Memory". In: *European Conference on Computer Vision (ECCV)*. 2018.

[34] Yanbei Chen, Xiatian Zhu, Wei Li, and Shaogang Gong. "Semi-supervised learning under class distribution mismatch". In: *AAAI Conference on Artificial Intelligence*. 2020.

[35] Hao Cheng, Zhaowei Zhu, Xingyu Li, Yifei Gong, Xing Sun, and Yang Liu. "Learning with Instance-Dependent Label Noise: A Sample Sieve Approach". In: *International Conference on Learning Representations (ICLR)*. 2021.

[36] Adam Coates, Andrew Ng, and Honglak Lee. "An Analysis of Single-Layer Networks in Unsupervised Feature Learning". In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*. 2011.

[37] Cody Coleman, Christopher Yeh, Stephen Mussmann, Baharan Mirzasoleiman, Peter Bailis, Percy Liang, Jure Leskovec, and Matei Zaharia. "Selection via proxy: Efficient data selection for deep learning". In: *International Conference on Learning Representations (ICLR)*. 2020.

[38] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. "Randaugment: Practical automated data augmentation with a reduced search space". In: *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRw)*. 2020.

[39] Ekin Dogus Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V. Le. "AutoAugment: Learning Augmentation Policies from Data". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[40] Bin Dai, Chen Zhu, Baining Guo, and David Wipf. "Compressing neural networks using the variational information bottleneck". In: *International Conference on Machine Learning (ICML)*. 2018.

[41] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Marc'aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, et al. "Large scale distributed deep networks". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2012.

[42] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. "ImageNet: A large-scale hierarchical image database". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2009.

[43] Terrance DeVries and Graham W Taylor. "Improved regularization of convolutional neural networks with cutout". In: *arXiv:1708.04552*. 2017.

[44] Y. Ding, L. Wang, D. Fan, and B. Gong. "A Semi-Supervised Two-Stage Approach to Learning from Noisy Labels". In: *IEEE Winter Conference on Applications of Computer Vision (WACV)*. 2018.

[45] Carl Doersch, Abhinav Gupta, and Alexei A Efros. "Unsupervised visual representation learning by context prediction". In: *International Conference on Computer Vision (ICCV)*. 2015.

[46] Carl Doersch and Andrew Zisserman. "Multi-Task Self-Supervised Visual Learning". In: *International Conference on Computer Vision (ICCV)*. 2017.

[47] Alexey Dosovitskiy, Philipp Fischer, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. "Discriminative unsupervised feature learning with exemplar convolutional neural networks". In: *IEEE transactions on pattern analysis and machine intelligence (TPAMI)*. 2016.

[48] John Duchi, Elad Hazan, and Yoram Singer. "Adaptive subgradient methods for online learning and stochastic optimization." In: *Journal of machine learning research*. 2011.

[49] Aysegul Dundar, Jonghoon Jin, and Eugenio Culurciello. "Convolutional clustering for unsupervised learning". In: *International Conference on Learning Representations (ICLR)*. 2015.

[50] Yanbo Fan, Siwei Lyu, Yiming Ying, and Baogang Hu. "Learning with average top-k loss". In: *Advances in Neural Information Processing systems (NeurIPS)*. 2017.

[51] B. Frenay and M. Verleysen. "Classification in the Presence of Label Noise: A Survey". In: *IEEE Transactions on Neural Networks and Learning Systems*. 2014.

[52] Theodoros Georgiou, Yu Liu, Wei Chen, and Michael Lew. "A survey of traditional and deep learning-based feature descriptors for high dimensional data in computer vision". In: *International Journal of Multimedia Information Retrieval*. 2020.

[53] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. "Unsupervised Representation Learning by Predicting Image Rotations". In: *International Conference on Learning Representations (ICLR)*. 2018.

[54] M. González, C. Bergmeir, I. Triguero, Y. Rodríguez, and J.M. Benítez. "Self-labeling techniques for semi-supervised time series classification: an empirical study". In: *Knowledge and Information Systems*. 2018.

[55] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. "Deep Learning". In: `http://www.deeplearningbook.org`. MIT Press, 2016, pp. 327–329.

[56] Siddharth Gopal. "Adaptive sampling for SGD by exploiting side information". In: *International Conference on Machine Learning*. 2016.

[57] Y. Grandvalet and Y. Bengio. "Semi-supervised Learning by Entropy Minimization". In: *International Conference on Neural Information Processing Systems (NIPS)*. 2004.

[58] Yves Grandvalet, Yoshua Bengio, et al. "Semi-supervised learning by entropy minimization"'. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2005.

[59] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, et al. "Bootstrap your own latent: A new approach to self-supervised learning". In: *Advances in neural information processing systems (NeurIPS)*. 2020.

[60] Sergio Guadarrama, Ryan Dahl, David Bieber, Jonathon Shlens, Mohammad Norouzi, and Kevin Murphy. "PixColor: Pixel Recursive Colorization". In: *British Machine Vision Conference (BMVC)*. 2017.

[61] Lan-Zhe Guo, Zhen-Yu Zhang, Yuan Jiang, Yu-Feng Li, and Zhi-Hua Zhou. "Safe deep semi-supervised learning for unseen-class unlabeled data". In: *International Conference on Machine Learning (ICML)*. 2020.

[62] S. Guo, W. Huang, H. Zhang, C. Zhuang, D. Dong, M.R. Scott, and D. Huang. "CurriculumNet: Weakly Supervised Learning from Large-Scale Web Images". In: *European Conference on Computer Vision (ECCV)*. 2018.

[63] Guy Hacohen and Daphna Weinshall. "On the power of curriculum learning in training deep networks". In: *International Conference on Machine Learning (ICML)*. 2019.

[64] B. Han, Q. Yao, X. Yu, G. Niu, M. Xu, W. Hu, I. Tsang, and M. Sugiyama. "Co-teaching: Robust training of deep neural networks with extremely noisy labels". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2018.

[65] Bo Han, Quanming Yao, Tongliang Liu, Gang Niu, Ivor W Tsang, James T Kwok, and Masashi Sugiyama. "A survey of label-noise representation learning: Past, present and future". In: *arXiv:2011.04406*. 2020.

[66] J. Han, P. Luo, and X. Wang. "Deep Self-Learning From Noisy Labels". In: *IEEE International Conference on Computer Vision (ICCV)*. 2019.

[67] Tengda Han, Weidi Xie, and Andrew Zisserman. "Self-supervised co-training for video representation learningg". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2020.

[68] H. Harutyunyan, K. Reing, G. V. Steeg, and A. Galstyan. "Improving Generalization by Controlling Label-Noise Information in Neural Network Weights". In: *International Conference on Machine Learning (ICML)*. 2020.

[69] Kaveh Hassani and Amir Hosein Khasahmadi. "Contrastive multi-view representation learning on graphs". In: *International Conference on Machine Learning (ICML)*. 2020.

[70] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. "Momentum Contrast for Unsupervised Visual Representation Learning". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[71] K. He, X. Zhang, S. Ren, and J. Sun. "Identity Mappings in Deep Residual Networks". In: *European Conference on Computer Vision (ECCV)*. 2016.

[72] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition". In: *IEEE conference on computer vision and pattern recognition (CVPR)*. 2016.

[73] D. Hendrycks, M. Mazeika, D. Wilson, and K. Gimpel. "Using Trusted Data to Train Deep Networks on Labels Corrupted by Severe Noise". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2018.

[74] D. Ho, E. Liang, X. Chen, I. Stoica, and P. Abbeel. "Population Based Augmentation: Efficient Learning of Augmentation Policy Schedules". In: *International Conference on Machine Learning (ICML)*. 2019.

[75] Daniel Ho, Eric Liang, Xi Chen, Ion Stoica, and Pieter Abbeel. "Population based augmentation: Efficient learning of augmentation policy schedules". In: *International Conference on Machine Learning*. 2019.

[76] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. "Mobilenets: Efficient convolutional neural networks for mobile vision applications". In: *arXiv:1704.04861*. 2017.

[77] Weihua Hu, Takeru Miyato, Seiya Tokui, Eiichi Matsumoto, and Masashi Sugiyama. "Learning discrete representations via information maximizing self-augmented training". In: *International Conference on Machine Learning (ICML)*. 2017.

[78] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. "Densely connected convolutional networks". In: *IEEE conference on computer vision and pattern recognition (CVPR)*. 2017.

[79] George Ioannou, Thanos Tagaris, and Andreas Stafylopatis. "Improving the Convergence Speed of Deep Neural Networks with Biased Sampling". In: *International Conference on Advances in Artificial Intelligence (ICAAI)*. 2019.

[80] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *International conference on machine learning (ICML)*. 2015.

[81] A. Iscen, G. Tolias, Y. Avrithis, and O. Chum. "Label Propagation for Deep Semi-supervised Learning". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[82] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. "Quantization and training of neural networks for efficient integer-arithmetic-only inference". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[83] Xu Ji, João F Henriques, and Andrea Vedaldi. "Invariant information clustering for unsupervised image classification and segmentation". In: *International Conference on Computer Vision (ICCV)*. 2019.

[84] Angela H Jiang, Daniel L-K Wong, Giulio Zhou, David G Andersen, Jeffrey Dean, Gregory R Ganger, Gauri Joshi, Michael Kaminksy, Michael Kozuch, Zachary C Lipton, et al. "Accelerating deep learning by focusing on the biggest losers". In: *arXiv:1910.00762*. 2019.

[85] L. Jiang, D. Huang, M. Liu, and W. Yang. "Beyond Synthetic Noise: Deep Learning on Controlled Noisy Labels". In: *International Conference on Machine Learning (ICML)*. 2020.

[86] L. Jiang, Z. Zhou, T. Leung, L.J. Li, and L. Fei-Fei. "MentorNet: Learning Data-Driven Curriculum for Very Deep Neural Networks on Corrupted Labels". In: *International Conference on Machine Learning (ICML)*. 2018.

[87] Tyler B Johnson and Carlos Guestrin. "Training deep models faster with robust, approximate importance sampling". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2018.

[88] Mohammad Kachuee, Orpaz Goldstein, Kimmo Karkkainen, Sajad Darabi, and Majid Sarrafzadeh. "Opportunistic learning: Budgeted cost-sensitive learning from data streams". In: *International Conference on Machine Learning (ICML)*. 2019.

[89] Yannis Kalantidis, Mert Bulent Sariyildiz, Noe Pion, Philippe Weinzaepfel, and Diane Larlus. "Hard negative mixing for contrastive learning". In: *Advances in neural information processing systems (NeurIPS)*. 2020.

[90] Guoliang Kang, Lu Jiang, Yi Yang, and Alexander G Hauptmann. "Contrastive adaptation network for unsupervised domain adaptation". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[91] Angelos Katharopoulos and François Fleuret. "Not all samples are created equal: Deep learning with importance sampling". In: *International Conference on Machine Learning (ICML)*. 2018.

[92] Kenji Kawaguchi and Haihao Lu. "Ordered SGD: A New Stochastic Optimization Framework for Empirical Risk Minimization". In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*. 2020.

[93] P. H. Le-Khac, G. Healy, and A. F. Smeaton. "Contrastive Representation Learning: A Framework and Review". In: *IEEE Access*. 2020.

[94] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan. "Supervised Contrastive Learning". In: *arXiv:2004.11362*. 2020.

[95] Y. Kim, J. Yim, J. Yun, and J. Kim. "NLNL: Negative Learning for Noisy Labels". In: *IEEE International Conference on Computer Vision (ICCV)*. 2019.

[96] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *International Conference on Learning Representations (ICLR)*. 2014.

[97] A. Kolesnikov, X. Zhai, and L. Beyer. "Revisiting Self-Supervised Visual Representation Learning". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[98] Alex Krizhevsky, Geoffrey Hinton, et al. *Learning multiple layers of features from tiny images*. Tech. rep. University of Toronto, 2009.

[99] Anders Krogh and John A Hertz. "A simple weight decay can improve generalization". In: *Advances in neural information processing systems (NeurIPS)*. 1992.

[100] S. Laine and T. Aila. "Temporal Ensembling for Semi-Supervised Learning". In: *International Conference on Learning Representations (ICLR)*. 2017.

[101] D.H. Lee. "Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks". In: *International Conference on Machine Learning Workshops (ICMLw)*. 2013.

[102] Kibok Lee, Yian Zhu, Kihyuk Sohn, Chun-Liang Li, Jinwoo Shin, and Honglak Lee. "i-Mix: A Domain-Agnostic Strategy for Contrastive Representation Learning". In: *International Conference on Learning Representations (ICLR)*. 2021.

[103] J. Li, R. Socher, and S.C.H. Hoi. "DivideMix: Learning with Noisy Labels as Semi-supervised Learning". In: *International Conference on Learning Representations (ICLR)*. 2020.

[104] Mengtian Li, Ersin Yumer, and Deva Ramanan. "Budgeted training: Rethinking deep neural network training under resource constraints". In: *International Conference on Learning Representations (ICLR)*. 2020.

[105] W. Li, L. Wang, W. Li, E. Agustsson, and L. Van Gool. "WebVision Database: Visual Learning and Understanding from Web Data". In: *arXiv: 1708.02862*. 2017.

[106] Y. Li, L. Liu, and R.T Tan. "Decoupled Certainty-Driven Consistency Loss for Semi-supervised Learning". In: *arXiv: 1901.05657*. 2019.

[107] Shaohui Lin, Rongrong Ji, Chenqian Yan, Baochang Zhang, Liujuan Cao, Qixiang Ye, Feiyue Huang, and David Doermann. "Towards Optimal Structured CNN Pruning via Generative Adversarial Learning". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[108] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. "Focal loss for dense object detection". In: *International Conference on Computer Vision (ICCV)*. 2017, pp. 2980–2988.

[109] Bo Liu, Haoxiang Li, Hao Kang, Nuno Vasconcelos, and Gang Hua. "Semi-supervised Long-tailed Recognition using Alternate Sampling". In: *arXiv:2105.00133*. 2021.

[110] S. Liu, J. Niles-Weed, N. Razavian, and C. Fernandez-Granda. "Early-Learning Regularization Prevents Memorization of Noisy Labels". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2020.

[111] Ilya Loshchilov and Frank Hutter. "Online batch selection for faster training of neural networks". In: *arXiv:1511.06343*. 2015.

[112] Haihao Lu and Rahul Mazumder. "Randomized gradient boosting machine". In: *arXiv:1810.10158*. 2018.

[113] Y. Luo, J. Zhu, M. Li, Y. Ren, and B. Zhang. "Smooth Neighbors on Teacher Graphs for Semi-Supervised Learning". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[114] X. Ma, H. Huang, Y. Wang, S. Romano, S. Erfani, and J. Bailey. "Normalized Loss Functions for Deep Learning with Noisy Labels". In: *International Conference on Machine Learning (ICML)*. 2020.

[115] X. Ma, Y. Wang, M.E. Houle, S. Zhou, S.M. Erfani, S.-T. Xia, S. Wijewickrema, and J. Bailey. "Dimensionality-Driven Learning with Noisy Labels". In: *International Conference on Machine Learning (ICML)*. 2018.

[116] Z. Ma and A. Leijon. "Bayesian Estimation of Beta Mixture Models with Variational Inference". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*. 2011.

[117] Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten. "Exploring the Limits of Weakly Supervised Pretraining". In: *European Conference on Computer Vision (ECCV)*. 2018.

[118] Kevin McGuinness. *Lecture slides in Data Analytics and Machine Learning*. `https://www101.dcu.ie/registry/module_contents.php?function=2&subcode=EE514`. 2021.

[119] L. McInnes, J. Healy, N. Saul, and L. Großberger. "UMAP: uniform manifold approximation and projection". In: *The Journal of Open Source Software*. 2018.

[120] Mikolov, Tomas and Sutskever, Ilya and Chen, Kai and Corrado, Greg and Dean, Jeffrey. "Distributed representations of words and phrases and their compositionality". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2013.

[121] George A Miller. *WordNet: An electronic lexical database*. MIT press, 1998.

[122] Baharan Mirzasoleiman, Jeff Bilmes, and Jure Leskovec. "Coresets for Data-efficient Training of Machine Learning Models". In: *International Conference on Machine Learning (ICML)*. 2020.

[123] T. Miyato, A.M. Dai, and I. Goodfellow. "Adversarial Training Methods for Semi-Supervised Text Classification". In: *arXiv: 1605.07725*. 2016.

[124] T. Miyato, S. Maeda, S. Ishii, and M. Koyama. "Virtual Adversarial Training: A Regularization Method for Supervised and Semi-Supervised Learning". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2018.

[125] Andriy Mnih and Koray Kavukcuoglu. "Learning word embeddings efficiently with noise-contrastive estimation". In: *Advances in neural information processing systems (NeurIPS)*. 2013.

[126] Jaemin Na, Heechul Jung, Hyung Jin Chang, and Wonjun Hwang. "FixBi: Bridging Domain Spaces for Unsupervised Domain Adaptation". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021.

[127] Feng Nan and Venkatesh Saligrama. "Adaptive classification for prediction under a budget". In: *Advances in neural information processing systems (NeurIPS)*. 2017.

[128] Loris Nanni, Stefano Ghidoni, and Sheryl Brahnam. "Handcrafted vs. non-handcrafted features for computer vision classification". In: *Pattern Recognition*. 2017.

[129] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A.Y. Ng. "Reading digits in natural images with unsupervised feature learning". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2011.

[130] D. T. Nguyen, C. K. Mummadi, T. P. N. Ngo, T. H. P. Nguyen, L. Beggel, and T. Brox. "SELF: Learning to Filter Noisy Labels with Self-Ensembling". In: *International Conference on Learning Representations (ICLR)*. 2020.

[131] Kien Nguyen, Clinton Fookes, Arun Ross, and Sridha Sridharan. "Iris recognition with off-the-shelf CNN features: A deep learning perspective". In: *IEEE Access*. 2017.

[132] Mehdi Noroozi and Paolo Favaro. "Unsupervised learning of visual representations by solving jigsaw puzzles". In: *European Conference on Computer Vision (ECCV)*. Springer. 2016.

[133] Niall O'Mahony, Sean Campbell, Anderson Carvalho, Suman Harapanahalli, Gustavo Velasco Hernandez, Lenka Krpalkova, Daniel Riordan, and Joseph Walsh. "Deep learning vs. traditional computer vision". In: *Science and Information Conference*. Springer. 2019, pp. 128–144.

[134] A. Oliver, A. Odena, C.A. Raffel, E.D. Cubuk, and I. Goodfellow. "Realistic Evaluation of Deep Semi-Supervised Learning Algorithms". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2018.

[135] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. "Representation learning with contrastive predictive coding". In: *arXiv:1807.03748*. 2018.

[136] D. Ortego, E. Arazo, P. Albert, N. O'Connor, and K. McGuinness. "Towards Robust Learning with Different Label Noise Distributions". In: *International Conference on Pattern Recognition (ICPR)*. 2020.

[137] D. Ortego, E. Arazo, P. Albert, N. O'Connor, and K. McGuinness. "Multi-Objective Interpolation Training for Robustness to Label Noise". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021.

[138] Kamalesh Palanisamy, Dipika Singhania, and Angela Yao. "Rethinking cnn models for audio classification". In: *arXiv:2007.11154*. 2020.

[139] Junting Pan, Cristian Canton Ferrer, Kevin McGuinness, Noel E O'Connor, Jordi Torres, Elisa Sayrol, and Xavier Giro-i-Nieto. "Salgan: Visual saliency prediction with generative adversarial networks". In: *arXiv:1701.01081*. 2017.

[140] Taesung Park, Alexei A Efros, Richard Zhang, and Jun-Yan Zhu. "Contrastive learning for unpaired image-to-image translation". In: *European Conference on Computer Vision (ECCV)*. 2020.

[141] G. Patrini, A. Rozza, A. Krishna Menon, R. Nock, and L. Qu. "Making Deep Neural Networks Robust to Label Noise: A Loss Correction Approach". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[142] H. Permuter, J. Francos, and I. Jermyn. "A study of Gaussian mixture models of color and texture features for image classification and segmentation". In: *Pattern Recognition*. 2006.

[143] Scott Plous. *The psychology of judgment and decision making*. Mcgraw-Hill Book Company, 1993.

[144] Rudra PK Poudel, Stephan Liwicki, and Roberto Cipolla. "Fast-scnn: fast semantic segmentation network". In: *arXiv:1902.04502*. 2019.

[145] Ning Qian. "On the momentum term in gradient descent learning algorithms". In: *Neural Networks : The Official Journal of the International Neural Network Society*. 1999.

[146] S. Qiao, W. Shen, Z. Zhang, B. Wang, and A. Yuille. "Deep Co-Training for Semi-Supervised Image Recognition". In: *European Conference on Computer Vision (ECCV)*. 2018.

[147] Siyuan Qiao, Huiyu Wang, Chenxi Liu, Wei Shen, and Alan Yuille. "Micro-Batch Training with Batch-Channel Normalization and Weight Standardization". In: *arXiv:1903.10520*. 2019.

[148] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. "Xnor-net: Imagenet classification using binary convolutional neural networks". In: *European Conference on Computer Vision (ECCV)*. 2016.

[149] S.-A. Rebuffi, S. Ehrhardt, K. Han, A. Vedaldi, and A. Zisserman. "Semi-Supervised Learning with Scarce Annotations". In: *International Conference on Computer Vision (ICCV)*. 2019.

[150] S. Reed, H. Lee, D. Anguelov, C. Szegedy, D. Erhan, and A. Rabinovich. "Training deep neural networks on noisy labels with bootstrapping". In: *International Conference on Learning Representations (ICLR)*. 2015.

[151] M. Ren, W. Zeng, B. Yang, and R. Urtasun. "Learning to Reweight Examples for Robust Deep Learning". In: *International Conference on Machine Learning (ICML)*. 2018.

[152] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. "Faster R-CNN: towards real-time object detection with region proposal networks". In: *IEEE transactions on pattern analysis and machine intelligence (TPAMI)*. 2016.

[153] N. Komodakis S. Zagoruyko. "Wide Residual Networks". In: *British Machine Vision Conference (BMVC)*. 2016.

[154] Ragav Sachdeva, Filipe R Cordeiro, Vasileios Belagiannis, Ian Reid, and Gustavo Carneiro. "Evidentialmix: Learning with combined open-set and closed-set noisy labels". In: *Winter Conference on Applications of Computer Vision (WACV)*. 2021.

[155] M. Sajjadi, M. Javanmardi, and T. Tasdizen. "Regularization With Stochastic Transformations and Perturbations for Deep Semi-Supervised Learning". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2016.

[156] T. Salimans and D.P. Kingma. "Weight Normalization: A Simple Reparameterization to Accelerate Training of Deep Neural Networks". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2016.

[157] Ozan Sener and Silvio Savarese. "Active learning for convolutional neural networks: A core-set approach". In: *International Conference on Learning Representations (ICLR)*. 2018.

[158] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. "CNN features off-the-shelf: an astounding baseline for recognition". In: *IEEE conference on computer vision and pattern recognition workshops (CVPRw)*. 2014.

[159] W. Shi, Y. Gong, C. Ding, Z. Ma, T. Xiaoyu, and N. Zheng. "Transductive Semi-Supervised Deep Learning using Min-Max Features". In: *European Conference on Computer Vision (ECCV)*. 2018.

[160] Connor Shorten and Taghi M Khoshgoftaar. "A survey on image data augmentation for deep learning". In: *Journal of Big Data*. Vol. 6. 1. Springer, 2019, pp. 1–48.

[161] Patrice Y Simard, David Steinkraus, John C Platt, et al. "Best practices for convolutional neural networks applied to visual document analysis." In: *International Conference on Document Analysis and Recognition (ICDAR)*. Vol. 3. 2003.

[162] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *International Conference on Learning Representations (ICLR)*. 2015.

[163] Leslie N Smith. "Cyclical learning rates for training neural networks". In: *IEEE Winter Conference on Applications of Computer Vision (WACV)*. 2017.

[164] Leslie N Smith and Nicholay Topin. "Super-convergence: Very fast training of neural networks using large learning rates". In: *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*. 2019.

[165] Kihyuk Sohn, David Berthelot, Chun-Liang Li, Zizhao Zhang, Nicholas Carlini, Ekin D Cubuk, Alex Kurakin, Han Zhang, and Colin Raffel. "Fixmatch:

Simplifying semi-supervised learning with consistency and confidence". In: *arXiv:2001.07685*. 2020.

[166] Hwanjun Song, Minseok Kim, Dongmin Park, and Jae-Gil Lee. "Learning from noisy labels with deep neural networks: A survey". In: *arXiv:2007.08199*. 2020.

[167] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research*. 2014.

[168] C. Stauffer and W. E. L. Grimson. "Adaptive background mixture models for real-time tracking". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 1999.

[169] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. "Deep High-Resolution Representation Learning for Human Pose Estimation". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[170] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. "Deep high-resolution representation learning for human pose estimation". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[171] Richard Sutton. "Two problems with back propagation and other steepest descent learning procedures for networks". In: *Proceedings of the Eighth Annual Conference of the Cognitive Science Society, 1986*. 1986.

[172] Ryo Takahashi, Takashi Matsubara, and Kuniaki Uehara. "Ricap: Random image cropping and patching data augmentation for deep cnns". In: *Asian Conference on Machine Learning (ACML)*. 2018.

[173] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. "Mnasnet: Platform-aware neural architecture search for mobile". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[174] Mingxing Tan and Quoc Le. "Efficientnet: Rethinking model scaling for convolutional neural networks". In: *International Conference on Machine Learning (ICML)*. 2019.

[175] D. Tanaka, D. Ikami, T. Yamasaki, and K. Aizawa. "Joint Optimization Framework for Learning with Noisy Labels". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[176] A. Tarvainen and H. Valpola. "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2017.

[177] S. Thulasidasan, T. Bhattacharya, J. Bilmes, G. Chennupati, and J. Mohd-Yusof. "Combating Label Noise in Deep Learning Using Abstention". In: *International Conference on Machine Learning (ICML)*. 2019.

[178] S. Thulasidasan, G. Chennupati, J. Bilmes, T. Bhattacharya, and S. Michalak. "On Mixup Training: Improved Calibration and Predictive Uncertainty for Deep Neural Networks". In: *arXiv: 1905.11001*. 2019.

[179] Sunil Thulasidasan, Gopinath Chennupati, Jeff A Bilmes, Tanmoy Bhattacharya, and Sarah Michalak. "On mixup training: Improved calibration and predictive uncertainty for deep neural networks". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.

[180] Y. Tian, C. Sun, B. Poole, P. Krishnan, C. Schmid, and P. Isola. "What Makes for Good Views for Contrastive Learning?" In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2020.

[181] Yonglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola. "What makes for good views for contrastive learning". In: *Advances in neural information processing systems (NeurIPS)*. 2020.

[182] Ruben Tolosana, Ruben Vera-Rodriguez, Julian Fierrez, Aythami Morales, and Javier Ortega-Garcia. "Deepfakes and beyond: A survey of face manipulation and fake detection". In: *Information Fusion*. 2020.

[183] Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J Gordon. "An empirical study of example forgetting during deep neural network learning". In: *International Conference on Learning Representations (ICLR)*. 2018.

[184] Hugo Touvron, Andrea Vedaldi, Matthijs Douze, and Hervé Jégou. "Fixing the train-test resolution discrepancy". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.

[185] D. Ulyanov, A. Vedaldi, and V. Lempitsky. "Deep Image Prior". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[186] Arash Vahdat. "Toward Robustness against Label Noise in Training Deep Discriminative Neural Networks". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2017.

[187] A. Veit, N. Alldrin, G. Chechik, I. Krasin, A. Gupta, and S. Belongie. "Learning From Noisy Large-Scale Datasets With Minimal Supervision". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[188] V. Verma, A. Lamb, J. Kannala, Y. Bengio, and D. Lopez-Paz. "Interpolation Consistency Training for Semi-Supervised Learning". In: *International Joint Conference on Artificial Intelligence (IJCAI)*. 2019.

[189] J. Vermorel and M. Mohri. "Multi-armed Bandit Algorithms and Empirical Evaluation". In: *European Conference on Machine Learning (ECML)*. 2005.

[190] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra. "Matching Networks for One Shot Learning". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2016.

[191] Qizhou Wang, Bo Han, Tongliang Liu, Gang Niu, Jian Yang, and Chen Gong. "Tackling Instance-Dependent Label Noise via a Universal Probabilistic Model". In: *Association for the Advancement of Artificial Intelligence (AAAI)*. 2021.

[192] Qizhou Wang, Jiangchao Yao, Chen Gong, Tongliang Liu, Mingming Gong, Hongxia Yang, and Bo Han. "Learning with group noise". In: *Association for the Advancement of Artificial Intelligence (AAAI)*. 2021.

[193] Tongzhou Wang and Phillip Isola. "Understanding contrastive representation learning through alignment and uniformity on the hypersphere". In: *International Conference on Machine Learning*. 2020.

[194] Y. Wang, W. Liu, X. Ma, J. Bailey, H. Zha, L. Song, and S.-T. Xia. "Iterative Learning With Open-Set Noisy Labels". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[195] Chen Wei, Kihyuk Sohn, Clayton Mellina, Alan Yuille, and Fan Yang. "CReST: A Class-Rebalancing Self-Training Framework for Imbalanced Semi-Supervised Learning". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021.

[196] H. Wei, L. Feng, X. Chen, and B. An. "Combating noisy labels by agreement: A joint training method with co-regularization". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[197] Daphna Weinshall, Gad Cohen, and Dan Amir. "Curriculum learning by transfer learning: Theory and experiments with deep networks". In: *International Conference on Machine Learning (ICML)*. 2018.

[198] Xiaoxia Wu, Ethan Dyer, and Behnam Neyshabur. "When Do Curricula Work?" In: *International Conference on Learning Representations (ICLR)*. 2021.

[199] Yuxin Wu and Kaiming He. "Group normalization". In: *European conference on computer vision (ECCV)*. 2018, pp. 3–19.

[200] Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. "Unsupervised feature learning via non-parametric instance discrimination". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2018.

[201] H. Zhang W. Huang M. R. Scott X. Wang. "Cross-Batch Memory for Embedding Learning". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[202] X. Xia, T. Liu, N. Wang, B. Han, C. Gong, G. Niu, and M. Sugiyama. "Are Anchor Points Really Indispensable in Label-Noise Learning?" In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.

[203] Xiaobo Xia, Tongliang Liu, Bo Han, Nannan Wang, Mingming Gong, Haifeng Liu, Gang Niu, Dacheng Tao, and Masashi Sugiyama. "Parts-dependent label noise: Towards instance-dependent label noise". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2020.

[204] T. Xiao, T. Xia, Y. Yang, C. Huang, and X. Wang. "Learning from massive noisy labeled data for image classification". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015.

[205] L. Xie, J. Wang, Z. Wei, M. Wang, and Q. Tian. "DisturbLabel: Regularizing CNN on the Loss Layer". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[206] Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V. Le. "Self-Training With Noisy Student Improves ImageNet Classification". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[207] Y. Xu, P. Cao, Y. Kong, and Y. Wang. "L_DMI: An Information-theoretic Noise-robust Loss Function". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.

[208] Guanglei Yang, Hao Tang, Zhun Zhong, Mingli Ding, Ling Shao, Nicu Sebe, and Elisa Ricci. "Transformer-Based Source-Free Domain Adaptation". In: *arXiv:2105.14138*. 2021.

[209] Yuzhe Yang and Zhi Xu. "Rethinking the value of labels for improving class-imbalanced learning". In: *Advances in neural information processing systems (NeurIPS)*. 2020.

[210] J. Yao, H. Wu, Y. Zhang, I.W Tsang, and J. Sun. "Safeguarded Dynamic Label Regression for Noisy Supervision". In: *Association for the Advancement of Artificial Intelligence Conference on Artificial Intelligence (AAAI)*. 2019.

[211] Ting Yao, Yingwei Pan, Yehao Li, and Tao Mei. "Exploring Visual Relationship for Image Captioning". In: *European Conference on Computer Vision (ECCV)*. 2018.

[212] Yazhou Yao, Zeren Sun, Chuanyi Zhang, Fumin Shen, Qi Wu, Jian Zhang, and Zhenmin Tang. "Jo-SRC: A Contrastive Approach for Combating Noisy Labels". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021.

[213] K. Yi and J. Wu. "Probabilistic End-To-End Noise Correction for Learning With Noisy Labels". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[214] X. Yu, B. Han, J. Yao, G. Niu, I. W. Tsang, and M. Sugiyama. "How does Disagreement Help Generalization against Label Corruption?" In: *International Conference on Machine Learning (ICML)*. 2019.

[215] Xiaohua Zhai, Avital Oliver, Alexander Kolesnikov, and Lucas Beyer. "S4l: Self-supervised semi-supervised learning". In: *International Conference on Computer Vision (ICCV)*. 2019.

[216] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. "Understanding deep learning requires re-thinking generalization". In: *International Conference on Learning Representations (ICLR)*. 2017.

[217] Cheng Zhang, Cengiz Öztireli, Stephan Mandt, and Giampiero Salvi. "Active mini-batch sampling using repulsive point processes". In: *AAAI Conference on Artificial Intelligence*. 2019.

[218] H. Zhang, M. Cisse, Y.N. Dauphin, and D. Lopez-Paz. "mixup: Beyond Empirical Risk Minimization". In: *International Conference on Learning Representations (ICLR)*. 2018.

[219] Jiong Zhang, Hsiang-Fu Yu, and Inderjit S Dhillon. "Autoassist: A framework to accelerate training of deep neural networks". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.

[220] Richard Zhang, Phillip Isola, and Alexei A Efros. "Colorful image colorization". In: *European conference on computer vision (ECCV)*. 2016.

[221] Richard Zhang, Phillip Isola, and Alexei A Efros. "Colorful image colorization". In: *European Conference on Computer Vision (ECCV)*. 2016.

[222] Z. Zhang, F. Ringeval, B. Dong, E. Coutinho, E. Marchi, and B. Schüller. "Enhanced semi-supervised learning for multimodal emotion recognition". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2016.

[223] Z. Zhang and M. Sabuncu. "Generalized Cross Entropy Loss for Training Deep Neural Networks with Noisy Labels". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2018.

[224] Junbo Jake Zhao, Michaël Mathieu, Ross Goroshin, and Yann LeCun. "Stacked What-Where Auto-encoders". In: *International Conference on Learning Representations workshop (ICLRw)*. 2015.

[225] Peilin Zhao and Tong Zhang. "Stochastic optimization with importance sampling for regularized loss minimization". In: *International Conference on Machine Learning (ICML)*. 2015.

[226] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. "Learning Deep Features for Discriminative Localization". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[227] Dengyong Zhou, Olivier Bousquet, Thomas N Lal, Jason Weston, and Bernhard Schölkopf. "Learning with local and global consistency". In: *Advances in neural information processing systems*. 2004.

[228] Huang Zhuo, Tai Ying, Wang Chengjie, Yang Jian, and Gong Chen. "They are Not Completely Useless: Towards Recycling Transferable Unlabeled Data for Class-Mismatched Semi-Supervised Learning". In: *arXiv:2011.13529*. 2020.

[229] Barret Zoph and Quoc V Le. "Neural architecture search with reinforcement learning". In: *International Conference on Learning Representations (ICLR)*. 2017.