DEEP LEARNING IN THE PRESENCE OF LABEL NOISE: A META-LEARNING APPROACH

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

GÖRKEM ALGAN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

MARCH 2021

Approval of the thesis:

DEEP LEARNING IN THE PRESENCE OF LABEL NOISE: A META-LEARNING APPROACH

submitted by **GÖRKEM ALGAN** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy in Electrical and Electronics Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** ______________

Prof. Dr. Ilkay Ulusoy
Head of Department, **Electrical and Electronics Engineering** ______________

Prof. Dr. Ilkay Ulusoy
Supervisor, **Electrical and Electronics Engineering, METU** ______________

Examining Committee Members:

Prof. Dr. Gözde Bozdağı Akar
Electrical and Electronics Engineering, METU ______________

Prof. Dr. Ilkay Ulusoy
Electrical and Electronics Engineering, METU ______________

Assoc. Prof. Dr. Selim Aksoy
Computer Engineering, Bilkent University ______________

Assist. Prof. Dr. Erdem Akagündüz
Electrical and Electronics Engineering, Cankaya University ______________

Assist. Prof. Dr. Elif Vural
Electrical and Electronics Engineering, METU ______________

Date: 12.03.2021

ABSTRACT

DEEP LEARNING IN THE PRESENCE OF LABEL NOISE: A META-LEARNING APPROACH

Algan, Görkem
Ph.D., Department of Electrical and Electronics Engineering
Supervisor: Prof. Dr. Ilkay Ulusoy

March 2021, 99 pages

Image classification systems recently made a giant leap with the advancement of deep neural networks. However, these systems require an excessive amount of labeled data to be adequately trained. Gathering a correctly annotated dataset is not always feasible due to practical challenges. Because of these practical challenges, label noise is a common problem in real-world datasets. This thesis presents two novel label noise robust learning algorithms: MSLG (Meta Soft Label Generation) and MetaLabelNet. Both algorithms are powered by meta-learning techniques and share the same learning framework. Proposed algorithms generate soft labels for each instance according to a meta-objective, which is to minimize the loss on the small meta-data. Afterward, the main classifier is trained on these generated soft-labels instead of given noisy labels. In each iteration, before conventional learning, the proposed meta objective reshapes the loss function so that resulting gradient updates would lead to model parameters with the minimum loss on meta-data. Different from MSLG, MetaLabelNet can work with dataset consists of both noisily labeled and unlabeled data, which is a problem setup that is not considered in the literature up to now. To prove the validity of the proposed algorithms, they are backed with mathematical justification. Extensive experiments on datasets with both synthetic and real-world label noises show the superiority of the proposed algorithms. For comparison with the state-of-the-art methods, proposed algorithms are tested on widely used noisily labeled benchmarking dataset Clothing1M. Both algorithms beat the baseline methods with a large margin, where MSLG achieves 2.3% and MetaLabelNet achieves 4.2% higher than the closest method. Results show that presented approaches are fully implementable for real-world use cases. Additionally, a novel label noise generation algorithm is presented for the purpose of generating realistic synthetic label noise.

Keywords: deep learning, label noise, noise robust, noise cleaning, meta-learning

ÖZ

KİRLİ ETİKETLERİN VARLIĞINDA DERİN ÖĞRENME: META-ÖĞRENİM YAKLAŞIMI

Algan, Görkem
Doktora, Elektrik ve Elektronik Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. Ilkay Ulusoy

Mart 2021 , 99 sayfa

Derin öğrenme algoritmalarının gelişmesi ile günümüzde bilgisayarlı görü teknolojilerinde büyük bir sıçrama yaşanmaktadır. Ancak yapay sinir ağlarını eğitmek için yüksek miktarda etiketlenmiş veri gerekmektedir. Veri setlerinin tamamıyla doğru etiketlenmesi çoğu zaman mümkün olmamaktadır. Bu tezde iki adet etiket kirliliğine karşı dayanıklı öğrenme algoritması önerilmiştir: MSLG (Meta Yumuşak Etiket Öğrenimi) ve MetaLabelNet. İki algoritma da meta-öğrenme tekniklerinden faydalanmakta ve aynı öğrenme sisteminden istifade etmektedir. Önerilen algoritmalar meta hedefe göre meta veri üzerinde kaybı en aza indirecek şekilde yumuşak etiketler üretir. Sonrasında ana sınıflandırıcı model kirli etieketler yerine üretilen yumuşak etiketler üzerinde eğitilir. Her eğitim adımında geleneksel makine öğreniminden önce, meta hedef model parametrelerinin en az kirlilikten etkilenecek şekilde güncellenmesine sebep olacak etiketleri üretir. Ayrıca MetaLabelNet hem kirli hem de etiketsiz verinin oluşturduğu veri setleri üzerinde de çalışabilmektedir. Bu problem türü literatürde daha önce çalışılmamıştır. Önerilen algoritmaların geçerliliği matematiksel olarak da kanıtlanmıştır. Metotların performansı hem sentetik hem de dünya kaynaklı gerçek kirliliğe sahip veri setleri üzerinde test edilmiştir. Sonuçlar önerilen algoritmaların yüksek başarısını doğrulamaktadır. Literatürdeki güncel algoritmalar ile performans kıyaslaması yapabilmek adına, önerilen algoritmalar bu alanda yaygın olarak kullanılan Clothing1M veri seti üzerinde test edilmiştir. İki algoritma da literatürdeki diğer metotların başarısının üstüne çıkmıştır. MSLG en başarılı metottan 2.3% fazla performans sağlarken MetaLabelNet 4.2% yüksek performans elde etmiştir. Sonuçlar algoritmaların gerçek dünya uygulamalarında kullanıma hazır olduğunu göstermektedir. Ayrıca tez kapsamında gerçekçi etiket kirliliği oluşturmak maksatlı da özgün bir algoritma önerilmiştir.

Anahtar Kelimeler: derin öğrenme, etiket kirliliği, gürültüye dayanıklı, kirlilik temizleme, meta-öğrenme

Dedicated to my beloved family...

ACKNOWLEDGMENTS

First of all, I would like to express my sincere gratitude to my supervisor Prof. Dr. Ilkay Ulusoy, for her continues support and patience during my PhD. thesis. Throughout this work, she always guided me to the right way whenever I faced with an obstacle. Without any doubt, her advices and wisdom are what made this thesis possible.

I would also like to thank Scientific and Technological Research Council of Turkey (TUBITAK) for their support with 2211-A National Scholarship Programme for PhD. students. Moreover, I would like to thank ASELSAN for its supportive attitude against PhD. students.

Lastly but undoubtedly most importantly, I thank to my family for raising me to who I am today. Feeling their continuous support during my whole life is the most precious fortune to me.

TABLE OF CONTENTS

ABSTRACT v

ÖZ vii

ACKNOWLEDGMENTS x

TABLE OF CONTENTS xi

LIST OF TABLES xvi

LIST OF FIGURES xvii

LIST OF ALGORITHMS xx

LIST OF ABBREVIATIONS xxi

CHAPTERS

1 INTRODUCTION 1

2 PRELIMINARIES 5

2.1 Problem Statement and Notations 5

2.2 Label Noise Models 6

2.3 Sources of Label Noise 7

2.4 Meta Learning 8

3 RELATED WORK 11

3.1 Noise Model Based Methods 12

3.1.1 Noisy Channel 12
3.1.1.1 Explicit calculation 15
3.1.1.2 Iterative calculation 16
3.1.1.3 Complex noisy channel 16
3.1.2 Label Noise Cleaning 17
3.1.2.1 Using data with clean labels 18
3.1.2.2 Using data with both clean and noisy labels 18
3.1.2.3 Using data with just noisy labels 18
3.1.3 Dataset Pruning 20
3.1.3.1 Removing Data 20
3.1.3.2 Removing Labels 21
3.1.4 Sample Choosing 21
3.1.4.1 Curriculum Learning 22
3.1.4.2 Multiple Classifiers 23
3.1.5 Sample Importance Weighting 24
3.1.6 Labeler Quality Assessment 26
3.1.7 Discussion 26
3.2 Noise Model Free Methods 27
3.2.1 Robust Losses 27
3.2.2 Meta Learning 29
3.2.3 Regularizers 30
3.2.4 Ensemble Methods 30
3.2.5 Others 31

3.2.6 Discussion 32
4 SYNTHETIC LABEL NOISE GENERATION 33
4.1 Methods from the Literature 33
4.1.1 Uniform Noise 34
4.1.2 Class-Dependent Noise 34
4.1.3 Feature-Dependent Noise 34
4.2 Proposed Noise Generation Algorithm 35
5 META SOFT LABEL GENERATION 39
5.1 Training 40
5.1.1 Training Phase-1 40
5.1.2 Training Phase-2 40
5.1.2.1 Meta Training Step 40
5.1.2.2 Conventional Training Step 41
5.2 Formulation of $\hat{y}$ 42
5.3 Reasoning of Classification Loss 44
5.4 Meta Objective 45
6 META-LABEL-NET 49
6.1 Training 49
6.1.1 Training Phase-1 50
6.1.2 Training Phase-2 50
6.1.2.1 Meta Training Step 50
6.1.2.2 Conventional Training Step 51
6.2 Learning with Unlabeled Data 52

6.3 Reasoning of Meta-Objective 52
7 EXPERIMENTS 57
7.1 CIFAR10 57
7.1.1 Dataset Description 58
7.1.2 Implementation Details 58
7.1.3 Results 58
7.2 Clothing1M 62
7.2.1 Dataset Description 62
7.2.2 Implementation Details 63
7.2.3 Results 64
7.3 Food101N 66
7.3.0.1 Dataset Description 66
7.3.1 Implementation Details 66
7.3.2 Results 66
7.4 WebVision 67
7.4.1 Dataset Description 67
7.4.2 Implementation Details 67
7.4.3 Results 68
7.5 Retinopaty of Prematurity 69
7.5.1 Dataset Description 69
7.5.2 Implementation Details 69
7.5.3 Results 70
8 CONCLUSION 73

REFERENCES 79

LIST OF TABLES

TABLES

Table 3.1 Existing noise model based methods to deal with label noise in the literature 13

Table 3.2 Existing noise model free methods to deal with label noise in the literature 14

Table 7.1 Hyper-parameters for CIFAR10 experiments for MSLG algorithm. . 59

Table 7.2 Test accuracies for CIFAR10 dataset with varying level of uniform noise. Results are averaged over 4 runs. 60

Table 7.3 Test accuracies for CIFAR10 dataset with varying level of feature-dependent noise. Results are averaged over 4 runs. 61

Table 7.4 Test accuracy percentages on Clothing1M dataset. All results are taken from the corresponding paper. 65

Table 7.5 Test accuracies for varying number of unlabeled data. Subset of Clothing1M dataset is used, which is balanced for each class. 65

Table 7.6 Test accuracy percentages for Food101N dataset. All values in the table are obtained from our own implementations. 67

Table 7.7 Test accuracies on WebVision dataset. Baseline results are taken from the [1] 68

Table 7.8 Train and test accuracies on ROP dataset. 71

LIST OF FIGURES

FIGURES

Figure 2.1 T-SNE plot of data distribution of MNIST dataset in feature space for 25% noise ratio. a) clean data b) random noise c) class-dependent noise which is still randomly distributed in feature domain d) feature-dependent noise in locally concentrated form e) feature-dependent noise that is concentrated on decision boundaries. 7

Figure 3.1 Noise can be modeled as a noisy channel on top of base classifier. Noisy channel adapts the characteristic of the noise so that base classifier is fed with noise-free gradients during traning. 15

Figure 3.2 Illustration of different types of algorithms. Starting from left; 1) representation of samples from a single class in the 2D space. Green samples represent the clean samples and red ones represent noisy samples. 2) label noise cleaning algorithms aim to correct the labels of noisy data 3) dataset pruning methods aim to eliminate noisy data (or just their labels) 4) sample importance weighting algorithms aim to up-weight clean samples and down-weight noisy samples (which is illustrated by size) 25

Figure 4.1 Data samples chosen by presented synthetic label noise generation algorithm. Left column shows the instances that are confidently classified, so their labels are not changed. On the other hand, right column shows ambiguous samples, for which labels are changed. 38

Figure 5.1 The overall framework of the proposed algorithm. x_n, y_n represents the training data and label pair. x_n^j, y_n^j represents batches of training data and label pair. x_m^j, y_m^j represents batches of meta-data and label pair. $\hat{y}$ represents generated soft-labels. The training consists of two phases. In the first phase (warm-up training), the base classifier is trained on noisy data with the conventional cross-entropy loss. This is useful to provide a stable starting point for the second phase. Training phase-2 consists of 3 consecutive steps, which are repeated for each batch of data. In the first iteration step, classification loss for model predictions is calculated with generated soft labels. Then, updated classifier parameters $\hat{\theta}$ are calculated by SGD. In the second iteration step, the cross-entropy loss for meta-data is calculated using the updated classifier parameters $\hat{\theta}$. Then, soft-labels are updated by taking an SGD step on this cross-entropy loss. In the third iteration step, the classifier parameters θ are updated by the classification loss (with newly generated soft labels) and entropy loss. 47

Figure 6.1 The overall framework of the proposed algorithm. x_n, y_n represents the training data and label pair. x_n^j, y_n^j represents batches of training data and label pair. x_m^j, y_m^j represents batches of meta-data and label pair. The training consists of two phases. In the first phase (warm-up training), the base classifier is trained on noisy data with the conventional cross-entropy loss. This is useful to provide a stable starting point for the second phase. Training phase-2 consists of 3 consecutive steps, which are repeated for each batch of data. In the first iteration step, classification loss for model predictions is calculated with soft labels produced by MetaLabelNet. Then, updated classifier parameters $\hat{\theta}$ are calculated by SGD. In the second iteration step, the cross-entropy loss for meta-data is calculated using the updated classifier parameters $\hat{\theta}$. Then, updated MetaLabelNet parameters $\hat{\phi}$ is calculated by taking an SGD step on this cross-entropy loss. In the third iteration step, new soft-labels are produced by updated MetaLabelNet parameters $\hat{\phi}$. Then, classifier parameters θ are updated by the classification loss (with newly generated soft labels) and entropy loss. Feature extractor is the pre-softmax output of the base classifier's duplicate trained at the warm-up phase of the training. 56

Figure 7.1 Test accuracies for different numbers of meta-data. 60

Figure 7.2 Colored lines represent the mean absolute difference among generated soft labels for consecutive epochs. Shaded regions are the variance of the differences. As can be seen MSLG generates highly unstabilized label predictions compared to MetaLabelNet. 63

Figure 7.3 Test accuracies of MetaLabelNet for different level of feature-dependent noises for varying sizes of unlabeled data. 64

LIST OF ALGORITHMS

ALGORITHMS

Algorithm 1 Generic Pipeline of Meta-Learning 9

Algorithm 2 Meta Soft Label Generation 43

Algorithm 3 Learning with MetaLabelNet 53

LIST OF ABBREVIATIONS

2D	2 Dimensional
3D	3 Dimensional
CCE	Categorical Cross Entropy
CNN	Convolutional Neural Networks
DNN	Deep Neural Networks
DR	Diabetic Retinopathy
FC	Fully Connected
MIL	Multiple Instance Learning
MLP	Multi Layer Perceptron
MSLG	Meta Soft Label Generation
NN	Neural Network
ROP	Retinopaty of Prematurity
SGD	Stochastic Gradient Descent
SLP	Single Layer Perceptron

CHAPTER 1

INTRODUCTION

Recent advancement in deep learning has led to great improvements on many different domains, such as image classification [2, 3], object detection [4, 5, 6], semantic segmentation [7, 8] and others. Despite their impressive ability to generalize, it is shown that these powerful models can memorize even complete random noise [9]. Various works are devoted to explaining this phenomenon [10], yet regularizing deep neural networks, while avoiding memorization, stays to be an important challenge. It gets even more crucial when there exists noise in data. Therefore, various methods are proposed in the literature to train deep neural networks effectively in the presence of noise.

There are two kinds of noise in literature, namely: feature noise and label noise [11]. Feature noise corresponds to corruption in the observed data features, while label noise means the change of label from its actual class. Even though both noise types may cause a significant decrease in the performance [12], label noise is considered to be more harmful [11] and shown to deteriorate the performance of classification systems in a broad range of problems [11, 13, 14]. This is due to several factors; the label is unique for each data while features are multiple, and the importance of each feature varies while the label always has a significant impact [12]. This work focuses on label noise; therefore, noise and label noise is used synonymously throughout the document.

The necessity of an excessive amount of labeled data for supervised learning is a significant drawback since it requires an expensive dataset collection and labeling process. To overcome this issue, cheaper alternatives have emerged. For example, an almost unlimited amount of data can be collected from the web via search engines or social media. Similarly, the labeling process can be crowdsourced with the help of systems like Amazon Mechanical Turk[1], Crowdflower[2], which decrease the cost of labeling notably. Another widely used approach is to label data with automated systems. However, all these approaches led to a common problem; label noise. Besides these methods, label noise can occur even in the case of expert annotators. Labelers may lack the necessary experience, or data can be too complex to be correctly classified even for the experts. Moreover, label noise can also be introduced to data for adversarial poisoning purposes [15, 16]. Being a natural outcome of dataset collection and labeling process makes label noise robust algorithms an essential topic for the development of efficient computer vision systems.

This thesis aims to develop a noise robust learning algorithm that is based on meta-learning techniques. Rest of the thesis is organized as follows. The thesis starts with providing essential information that is required for the rest of the document, in Chapter 2. This chapter starts with the formal definition of the learning from noisy labels problem. Afterward, label noise types are introduced. Even though the main purpose of this thesis is to develop a label noise robust learning algorithm, it is beneficial to understand the cause of the problem. Therefore, causes of the noisy labels are also investigated in this chapter.

Supervised learning with label noise is an old phenomenon with three decades of history [17]. Even though deep networks are considered to be relatively robust to label noise, they have an immense capacity to overfit data [9]. Therefore, preventing DNNs to overfit noisy data is very important, especially for fail-safe applications, such as automated medical diagnosis systems. Considering the significant success of deep learning over its alternatives, it is a topic of interest, and many works are presented in the literature. Chapter 3 presents a detailed analyses of the proposed methods from the literature. For a clear overview of the literature, proposed algorithm are grouped in two major groups as noise model based and noise model free. Algorithms in the first group aim to estimate the structure of the noise and use this information to avoid the negative effects of noisy labels during training. On the other hand, methods in the second group try to come up with algorithms that are inherently noise robust. More-

[1] http://www.mturk.com
[2] http://crowdflower.com

over, proposed algorithms are further sub-grouped according to their logic behind the proposed algorithms.

In order to develop label noise robust algorithm, a test set with verified clean labels is required to be able to evaluate performance of the proposed system. Most of the real-world noisy datasets [18, 19] have a small number of verified data to be used as test set. Even though these real-world noisy datasets are useful for benchmarking purposes, they are not suitable for development purposes. There are two reasons for that. Firstly, these datasets are commonly collected from the web and they are massive. Therefore, they require large amount of computation even for a single epoch, which slows down the algorithm development phase. Secondly, since these datasets are collected from the web, their noise ratio is not known for sure. More importantly, noise ratio is fixed so that one can not test the proposed algorithm for different noise rates. As a result, for quick test and deployment, conventional methodology is to add synthetic label noise to toy datasets and develop the algorithm on this dataset. After the initial proof of the concept, algorithm can be run on a real-world dataset. However, if the introduced artificial noise is not realistic, then it would cause an undesired solution which is not applicable to problem domain. Therefore, a novel methodology to generate realistic feature-dependent noisy labels is presented in Chapter 4.

A label noise robust learning methodology is proposed in Chapter 5. Proposed algorithm uses meta-learning paradigms to iteratively generate soft-labels for each instance, so that resulting gradient steps would lead in the direction of most noise robust model parameters. This is achieved by a meta learning loop that is processed before conventional training for each batch of data. Firstly soft-labels are generated in meta training loop, and secondly base classifier model is trained to match these soft-labels. Proposed algorithm is based on the following simple assumption: *optimal model parameters learned with noisy training data should minimize the cross-entropy loss on clean meta-data*. This algorithm is named as MSLG (**M**eta **S**oft **L**abel **G**eneration).

Even though MSLG achieves state of the art results on datasets with noisy labels, there is still room for improvement. In the proposed method, label predictions are formulated as free differentiable parameters that does not depend on data features. However, there is a certain correlation between data features and labels. Using this information is beneficial to predict more accurate soft labels. Therefore, proposed methods is further utilized in Chapter 6. In this improved version, soft labels are predicted by an SLP network that is called as MetaLabelNet. MetaLabelNet gets data features as input and weights are updated via meta-loss. As a result, different from MSLG, there is no soft-label prediction corresponding to each instance, but rather a generic soft-label generation algorithm that depends on data features. Since labels are not free differentiable variables, which changes rapidly throughout the training, MetaLabelNet framework provides much stable soft-labels which is advantageous for effective learning. Moreover, MetaLabelNet framework is independent of training data labels, so it can be applied to unlabeled data in the same way as labeled data. Therefore, on top of noisily labeled data, the proposed framework can work with unlabeled data too. To the best of the knowledge of the author, there is no work in the literature that is able to work with both unlabeled and noisily labeled data.

Chapter 7 presents the experimental results for the proposed algorithms with various datasets. First, proposed algorithms are tested on CIFAR10 dataset [20] with synthetic noise. Afterward, to show the effectiveness of the presented methods, they are tested on two real-world noisy datasets WebVision [19] and Clothing1M [18]. These datasets are widely accepted as benchmarking datasets by the literature. Results show that both MSLG and MetaLabelNet achieves state of the art performance on the given datasets. Finally, proposed methods are tested on a case study of autonomous diagnosis of ROP (Retinopathy of Prematurity Plus) disease. This dataset is labeled by three different experts and contains extreme level of label noise due to conflicting opinions of annotators. Learning objective is to correctly classify retina images into stages of ROP disease. Results show the practical usability of the proposed solutions, where they achieve higher accuracy than classical learning with cross entropy loss.

Finally, Chapter 8 concludes the thesis with briefly summarizing the works done in the thesis and providing further discussion on the topic.

CHAPTER 2

PRELIMINARIES

In this section, firstly the problem statement for supervised learning in the presence of noisy labels is given. Secondly, types of label noises are presented. Thirdly, sources of label noise are discussed. Finally, preliminary knowledge on meta-learning is provided for better understanding of the proposed algorithms in Chapter 5 and Chapter 6.

2.1 Problem Statement and Notations

In supervised learning we have a clean dataset $\mathcal{S} = \{(x_1, y_1), ..., (x_N, y_N)\} \in (X, Y^h)$ drawn according to an unknown distribution $\mathcal{D}$, over (X, Y^h) where X represents the feature space and Y^h represents the hard-label space for which each label is encoded into one-hot vector. The aim is to find the best mapping function $f : X \rightarrow Y^h$ that is parametrized by θ.

$$\theta^\star = \arg\min_\theta R_{l,\mathcal{D}}(f_\theta) \tag{2.1}$$

where $R_{l,\mathcal{D}}$ is the empirical risk defined for loss function l and distribution $\mathcal{D}$. In the presence of the noise, dataset turns into $\mathcal{S}_n = \{(x_1, y_{n,1}), ..., (x_N, y_{n,N})\} \in (X, Y^h)$ drawn according to a noisy distribution $\mathcal{D}_n$, over (X, Y^h). Then, risk minimization results in a different parameter set $\theta_n^\star$.

$$\theta_n^\star = \arg\min_\theta R_{l,\mathcal{D}_n}(f_\theta) \tag{2.2}$$

where $R_{l,\mathcal{D}_n}$ is the empirical risk defined for the same loss function l and noisy distribution $\mathcal{D}_n$.

Classical supervised learning (Equation 2.3) aims to find the best estimator parameters $\theta^\star$ for given distribution $\mathcal{D}$ while iterating over $\mathcal{D}$. However, in noisy label setup, the task is still finding $\theta^\star$ while working on distribution $\mathcal{D}_n$. Therefore, classical risk minimization is insufficient in the presence of label noise, since it would result in $\theta_n^\star$. As a result, variations of classical risk minimization methods are proposed in the literature, and they will be further evaluated in the upcoming sections.

$$\theta^{(t+1)} = \theta^{(t)} - \lambda \nabla_\theta l(f_\theta(x), y_n) \tag{2.3}$$

Throughout this thesis, $\mathcal{D}_n$ represents the noisy training data distribution, $\mathcal{D}_m$ represents the meta-data distribution, and $\hat{\mathcal{D}}$ represents the distribution of training data with generated soft-labels. Both $\mathcal{D}$ and $\mathcal{D}_n$ is defined over hard label space Y^h, while $\hat{\mathcal{D}}$ is defined over soft label space Y^s. Soft labels encode more information since non-zero probability values are assigned for each class. N is the number of training data samples, and M is the number of meta-data samples, where $M << N$. N_b represents the number of data in each batch, and it is the same for both training data and meta-data. We represent the number of classes as C, and superscript represents the label probability for that class, such that y_i^j represents the label value of i^{th} sample for class j.

2.2 Label Noise Models

A detailed taxonomy of label noise is provided in [12]. This work follows the same taxonomy with a little abuse of notation. Label noise can be affected by three factors: data features, the true label of data, and the labeler characteristics. According to the dependence of these factors, label noise can be categorized into three subclasses.

Uniform noise is totally random and depends on neither instance features nor its true class. With a given probability p_e label is changed from its true class. *Class-dependent noise* is independent of image features but depends on its class; $p_e = p(e|y)$. That means data from a particular class are more likely to be mislabeled. For example, in a handwritten digit recognition task, "3" and "8" are much more likely to be confused with each other rather than "3" and "5". *Feature-dependent noise* depends on both image features and its class; $p_e = p(e|x, y)$. As in the class-dependent case, objects from a particular class may be more likely to be mislabeled. Moreover, the chance of mislabeling may change according to data features. If an instance has similar features to another instance from another class, it is more likely to be mislabeled. Generating xy-dependent synthetic noise is harder than the previous two models; therefore, some works tried to provide a generic framework by either checking the complexity of data [21] or their position in feature space [22]. All these types of noises are illustrated in Figure 2.1

The case of multi-labeled data, in which each instance has multiple labels given by different annotators, is not considered here. In that scenario, works show that modeling each labeler's characteristics and using this information during training significantly boosts the performance [23]. However, various characteristics of different labelers can be explained with given noise models. For example, in a crowd-sourced dataset, some labelers can be total spammers who label with a random selection [24]; therefore, they can be modeled as random noise. On the other hand, labelers with better accuracies than random selection can be modeled by class-dependent or feature-dependent noise. As a result, the labeler's characteristic is not introduced as an extra ingredient in these definitions.

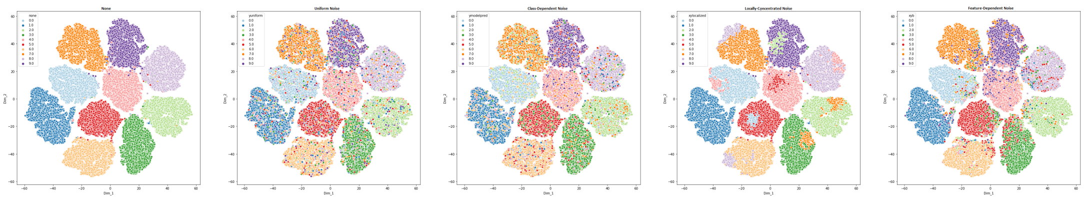


Figure 2.1: T-SNE plot of data distribution of MNIST dataset in feature space for 25% noise ratio. a) clean data b) random noise c) class-dependent noise which is still randomly distributed in feature domain d) feature-dependent noise in locally concentrated form e) feature-dependent noise that is concentrated on decision boundaries.

2.3 Sources of Label Noise

As mentioned, label noise is a natural outcome of dataset collection process and can occur in various domains, such as medical imaging [25, 26, 23], semantic segmentation [27, 28, 29], crowd-sourcing [30], social network tagging [31, 32], financial analysis [33] and many more. This work focuses on various solutions to such problems, but it may be helpful to investigate the causes of label noise to better understand the phenomenon.

Firstly, with the availability of the immense amount of data on the web and social media, it is a great interest of computer vision community to make use of that [34, 35, 36, 37, 38, 39]. Nevertheless, labels of these data are coming from messy user tags or automated systems used by search engines. These processes of obtaining datasets are well known to result in noisy labels.

Secondly, the dataset can be labeled by multiple experts resulting in a multi-labeled dataset. Each labeler has a varying level of expertise, and their opinions may commonly conflict with each other, which results in noisy label problem [24]. There are several reasons to get data labeled by more than one expert. Opinions of multiple labelers can be used to double-check each other's predictions for challenging datasets, or crowd-sourcing platforms can be used to decrease the cost of labeling for big data. Despite its cheapness, labels obtained from non-experts are commonly noisy with a differentiating rate of error. Some labelers even can be a total spammer who labels with random selection [24].

Thirdly, data can be too complicated for even the experts in the field, e.g., medical imaging. For example, to collect gold standard validation data for retinal images, annotations are gathered from 6-8 different experts [40, 41]. This can be due to the subjectiveness of the task for human experts or the lack of annotator experience. Considering the fields where the accurate diagnosis is of crucial importance, overcoming this noise is of great interest.

Lastly, label noise can intentionally be injected in purpose of regularizing [42] or data poisoning [15, 16].

2.4 Meta Learning

With the recent advancements of deep neural networks, the necessity of hand-designed features for computer vision systems are mostly eliminated. Instead, these features are learned autonomously via machine learning techniques. Even though these algorithms can learn complex functions on their own, there remain many hand-designed parameters such as network architecture, loss function, optimizer algorithm, and so on. Meta-learning aims to eliminate these necessities by learning not just the required complex function for the task, but also learning the learning itself [43, 44].

Meta-learning aims to utilize the learning for a meta task, which is a higher-level learning objective than classical supervised learning. Meta task can be from a wide variety of a learning objectives such as; knowledge transfer [45], neural architecture search [46], optimizer optimization [43], optimal weight initialization [44] and etc.

In general, meta-learning framework consists of two training loops that are repeated consecutively. First training loop is the conventional training where base model is trained with traditional deep learning methods. Second loop is the meta-training loop, for which the task is to optimize the conventional training loop via meta-parameters. Meta-parameters can be any hyperparameter that has an impact on the conventional training loop. For meta training loop to optimize conventional training loop, it should receive feedback from the base classifier. For this purpose, meta-training loop repeats the conventional training and finds updated parameters for the network. Then back-propagating through the updated model parameters, it seeks for the optimal meta-parameter set. This is more like asking the question *"what should have been the meta-parameters so that model trained with them would minimize the meta learning objective"*. After updating meta parameters, model is trained in conventional training manner with the meta-parameters. Generic pipeline for the meta-learning approaches are given in Algorithm 1.

Algorithm 1: Generic Pipeline of Meta-Learning

```
Input: Model parameters $\theta$, Meta-parameters $\phi$
Output: Updated model parameters $\theta$, Updated meta-parameters $\phi$
while not finished do
    Find updated model parameters $\hat{\theta} = F(\phi, \theta)$;
    Update meta-parameters $\phi = M(\hat{\theta}, \phi)$;
    Update model parameters $\theta = F(\phi, \theta)$;
end
```

Meta-learning is a recently evolving field that can be applied to various fields. One of the most successfully implementation of the meta-learning paradigm is MAML [44], which achieved the state of the art results in many various fields. Aiming to find optimal network parameters for a few-shot learning task, MAML seeks optimal weight initialization by taking gradient steps on the meta objective. Meta objective is defined as to minimize the loss on an ensemble of tasks. Therefore, network is forced to figure out most generic weights that can easily fine tuned for a specific task in a few epochs of training.

A major drawback for the applicability of meta-learning is its computational cost. It requires two nested loops of training. Moreover, it requires gradients of the gradients for the meta objective, which needs second order derivative computation. As a result, meta-learning based methods are computationally demanding. However, this extra computational cost is getting less preventive due to the advancements in the hardware technologies. Therefore, meta-learning based algorithms have a big potential for the overcoming challenges in the future.

CHAPTER 3

RELATED WORK

There are many possible ways to group proposed methods in the literature. For example, one possible way to distinguish algorithms is according to their need for a noise-free subset of data or not. Alternatively, they can be divided according to the noise type they are dealing with, or label type such as singly-labeled or multi-labeled. However, these are not handy to understand the main approaches behind the proposed algorithms; therefore, different sectioning is proposed as noise model based and noise model free methods.

Noise model based methods aim to model the noise structure so that this information can be used during training to come through noisy labels. In general, approaches in this category aim to extract noise-free information contained within the dataset by either neglecting or de-emphasizing information coming from noisy samples. Furthermore, some methods attempt to reform the dataset by correcting noisy labels to increase the quality of the dataset for the classifier. The performance of these methods is heavily dependent on the accurate estimate of the underlying noise. The advantage of noise model based methods is the decoupling of classification and label noise estimation, which helps them to work with the classification algorithm at hand. Another good side is in the case of prior knowledge about the noise structure, noise model based methods can easily be head-started with this extra information inserted to the system.

Differently, noise model free methods aim to develop inherently noise robust strategies without explicit modeling of the noise structure. These approaches assume that classifier is not too sensitive to the noise, and performance degradation results from overfitting. Therefore, the main focus is given to overfit avoidance by regularizing the network training procedure.

Both of the mentioned approaches are discussed and further categorized in section 3.1 and section 3.2. Table 3.1 and Table 3.2 presents all the mentioned methods to provide a clear picture as a whole. It should be noted that most of the time there are no sharp boundaries among the algorithms, and they may belong to more than one category. However, for the sake of integrity, they are placed in the subclass of most resemblance. The content of this chapter is previously published in [47] and the content is highly correlated with the original paper.

3.1 Noise Model Based Methods

In the presence of noisy labels, the task is to find the best estimator for hidden distribution $\mathcal{D}$, while iterating over distribution $\mathcal{D}_n$. If the mapping function $M : \mathcal{D} \rightarrow \mathcal{D}_n$ is known, it can be used to reverse the effect of noisy samples. Algorithms under this section simultaneously try to find underlying noise structure and train the base classifier with estimated noise parameters. They need a better estimate of M to train better classifiers and better classifiers to estimate M accurately. Therefore, they usually suffer from a chicken-egg problem. Approaches belonging to this category are explained in the following subsections.

3.1.1 Noisy Channel

The general setup for the noisy channel is illustrated in Figure 3.1. Methods belonging to this category minimize the following risk

$$\hat{R}_{l,\mathcal{D}}(f) = \frac{1}{N}\sum_{i=1}^{N} l(Q(f_\theta(x_i)), \tilde{y_i}) \tag{3.1}$$

where $Q(f_\theta(x_i)) = p(\tilde{y_i}|f_\theta(x_i))$ is the mapping from network predictions to given noisy labels. If Q adapts the noise structure $p(\tilde{y}|y)$, then network will be forced to learn true mapping $p(y|x)$.

Q can be formulated with a *noise transition matrix* T so that $Q(f_\theta(x_i)) = Tf_\theta(x_i)$

Table 3.1: Existing noise model based methods to deal with label noise in the literature

Noise Model Based Methods	**1. Noisy Channel** *a.Explicit calculation*: noisy data [48], clean data [49], easy data [50] *b.Iterative calculation*: EM [51, 52, 25], FC-layer [53, 54] anchor point [55], Drichlet-distribution [56] *c.Complex noisy channel*: noise type [18], relevance [57]
	2. Label Noise Cleaning *a.Using data with clean labels*: clean set [58], ensemble [59] graph-based [60] *b.Using data with both clean and noisy labels*: iteratively [61], [62] *c.Using data with just noisy labels*: posterior [63], compatibility [64] consistency [65, 66, 67], ensemble [68], prototypes [69] quality embedding [70], partial labels [71]
	3. Dataset Pruning *a.Removing Data* prediction [72], ensemble [73, 74] noise rate [75], transfer learning [76], cyclic state [77], K-means [78] *b.Removing Labels* SSL [79, 80, 81, 1], relabeling [82, 83]
	4. Sample Choosing a.*Curriculum Learning*: Loss [84], teacher-student [85] uncertain samples [86], curriculum loss [87], complexity [88] consistency [89] b.*Multiple Classifiers*: Consistency [90], co-teaching [91, 92, 93, 94]
	5. Sample Importance Weighting Meta task [95, 96, 97], siamese network [98], pLOF [26] abstention [99], noise rate [100, 101], similarity loss [102] transfer learning [103], θ-distribution [104]
	6. Labeler Quality Assessment EM [105, 106, 24], trace regularizer [107], crowd-layer [108] difficulty estimate [109], consistency [110], omitting variable [111] softmax layer per labeler [23]

Table 3.2: Existing noise model free methods to deal with label noise in the literature

Noise Model Free Methods	**1. Robust Losses** Non-convex [112, 113, 114], 0-1 surrogate [115], MAE [116] IMEA [117], Generalized cross-entropy [118], symmetric loss [119] unbiased estimator [120], modified cce [121], information loss [122] linear-odd [123], classification calibrated [124], SGD [125]
	2. Meta Learning Choosing best methods [126], pumpout [127], parameter initialization [128], knowledge distillation [129], gradient magnitude adjustment [130, 131] meta soft labels [132]
	3. Regularizers Dropout [133], adversarial training [134], mixup [135] label smoothing [136, 137], pre-training [138], dropout on final layer [139] checking dimensionality [140], auxiliary image regularizer [141]
	4. Ensemble Methods LogitBoost&BrownBoost [142], noise detection based AdaBoost [143] rBoost [144], RBoost1&RBoost2 [145], robust multi-class AdaBoost [146]
	5. Others Complementary labels [147, 148], auto-encoder [149], less noisy data [150], data quality [151], prototype learning [152, 153], MIL [154, 155]

where each element of the matrix represents the transition probability of given true label to noisy label, $T_{ij} = p(\tilde{y} = j|y = i)$. Since T is composed of probabilities, weights coming from a single node should sum to one $\sum_j T_{ij} = 1$. This procedure of correcting predictions to match given label distribution is also called *loss-correction* [48].

A common problem in noisy channel estimation is scalability. As the number of classes increases, the size of the noise transition matrix increases exponentially, making it intractable to calculate. This can be partially avoided by allowing connections only among the most probable nodes [52] or predefined nodes [156]. These restrictions are determined by human experts, which allows additional noise information to be inserted into the training procedure.

The noisy channel is used only in the training phase. In the evaluation phase, the noisy channel is simply removed to get noise-free predictions of the base classifier. In these kinds of approaches, performance heavily depends on the accurate estimation of noisy channel parameters; therefore, works mainly focus on the estimation of Q. Various ways of formulating the noisy channel are explained below.

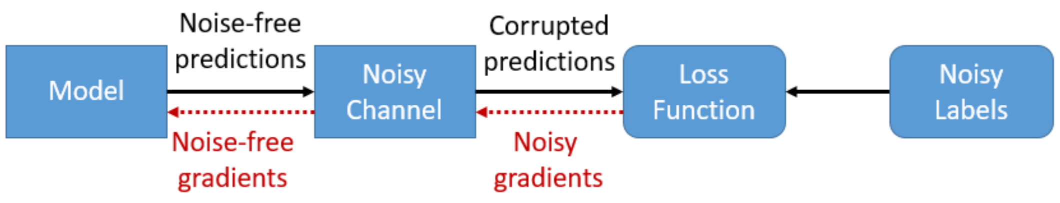


Figure 3.1: Noise can be modeled as a noisy channel on top of base classifier. Noisy channel adapts the characteristic of the noise so that base classifier is fed with noise-free gradients during traning.

3.1.1.1 Explicit calculation

Noise transition matrix is calculated explicitly, and then the base classifier is trained using this matrix. Assuming dataset is balanced in terms of clean representative samples and noisy samples, so that there exists samples for each class with $p(y = \tilde{y}_i|x_i) = 1$, [48] constructs T just based on noisy class probability estimates of a pre-trained model, so-called *confusion matrix*. A similar approach is followed in [49]; however, the noise transition matrix is calculated from the network's confusion matrix on the clean subset of data. Two datasets are gathered in [50], namely: easy data and hard data. The classifier is first trained on the easy data to extract similarity relationships among classes. Afterward, the calculated similarity matrix is used as the noise transition matrix. Another method proposed in [53] calculates the confusion matrix on both noisy data and clean data. Then, the difference between these two confusion matrices gives T.

3.1.1.2 Iterative calculation

Noise transition matrix is estimated incrementally during the training of the base classifier. In [51, 52] expectation-maximization (EM) [157] is used to iteratively train network to match given distribution and estimate noise transition matrix given the model prediction. The same approach is used on medical data with noisy labels in [25]. [54] and [53] add a linear fully connected layer as a last layer of the base classifier, which is trained to adapt noise behavior. In order to avoid this additional layer to converge the identity matrix and base classifier overfitting the noise, the weight decay regularizer is applied to this layer. [55] suggests using class probability estimates on anchor points (data points that belong to a specific class almost surely) to construct the noise transition matrix. In the absence of a noise-free subset of data, anchor points are extracted from data points with high noisy class posterior probabilities. Then, the matrix is updated iteratively to minimize loss during training. Instead of using softmax probabilities, [56] models noise transition matrix in Bayesian form by projecting it into a Dirichlet-distributed space.

3.1.1.3 Complex noisy channel

Different then simple confusion matrix, some works formalize the noisy channel as a more complex function. This enables noisy channel parameters to be calculated not just by using network outputs, but additional information about the content of data. For example, three types of label noises are defined in [18], namely: no noise, random noise, structured noise. An additional convolutional neural network (CNN) is used to interpret the noise type of each sample. Finally, the noisy layer aims to match predicted labels to noisy labels with the help of predicted noise type. Another work in [57] proposes training an extra network as a relevance estimator, which attains the label's relevance to the given instance. Predicted labels are mapped to noisy labels with the consideration of relevance. If relevance is low, in case of noise, the classifier can still make predictions of true class and doesn't get penalized much for it.

3.1.2 Label Noise Cleaning

An obvious solution to noisy labels is to identify and correct suspicious labels to their corresponding true classes. Cleaning the whole dataset manually can be costly; therefore, some works propose to pick only suspicious samples to be sent to a human annotator for the purpose of reducing the cost [39]. However, this is still not a scalable approach. As a result, various algorithms are proposed in the literature. Including the label correction algorithm, the empirical risk takes the following form

$$\hat{R}_{l,\mathcal{D}}(f) = \frac{1}{N}\sum_{i=1}^{N} l(f_\theta(x_i), G(\tilde{y}_i, x_i)) \tag{3.2}$$

where $G(\tilde{y}_i, x_i) = p(y_i|\tilde{y}_i, x_i)$ represents the label cleaning algorithm. Label cleaning algorithms rely on a feature extractor to map data to feature domain for the investigation of noisiness. While some works use a pretrained network as the feature extractor, others use the base classifier as it gets more and more accurate during training. This results in an iterative framework: as the classifier gets better the label cleaning is more accurate, and as the label quality gets better the classifier gets better. From this point of view, label cleaning can be viewed as a dynamically evolving component of the system, instead of the preprocessing data. Such methods usually tackle the difficulty of distinguishing informative hard samples from those with noisy labels [12]. As a result, they can end up removing too many samples or changing labels in a delusional way. Approaches for label cleaning can be separated according to their need for clean data or not.

3.1.2.1 Using data with clean labels

In the existence of a clean subset of data, the aim is to fuse noise-free label structure to noisy labels for correction. If the clean subset is large enough to train a network, one obvious way is to relabel noisy labels by predictions of the network trained on clean data. For relabeling, [58] uses alpha blending of given noisy labels and predicted labels. An ensemble of networks that are trained with different subsets of the dataset is used in [59]. If they all agree on the label, it is changed to the predicted label; otherwise, the label is set to a random label. Instead of keeping the noisy label, setting it randomly helps to break the structure in noise and makes noise more uniformly distributed in label space. In [60] a graph-based approach is used, where relation among noisy labels and clean labels are extracted by a conditional random field.

3.1.2.2 Using data with both clean and noisy labels

Some works rely on a subset of data, for which both clean and noisy labels are provided. Then label noise structure is extracted from these conflicting labels and used to correct noisy data. In [61], the label cleaning network gets two inputs: extracted features of instances by the base classifier and corresponding noisy labels. Label cleaning network and base classifier are trained jointly, so that label cleaning network learns to correct labels on the clean subset of data and provides corrected labels for base classifier on noisy data. Same approach is decoupled in [62] in teacher-student manner. First, the student is trained on noisy data. Then features are extracted from the clean data via the student model, and the teacher learns the structure of noise depending on these extracted features. Afterward, the teacher predicts soft labels for noisy data, and the student is again trained on these soft labels for fine-tuning.

3.1.2.3 Using data with just noisy labels

Noise-free data is not always available, so the main approach in this situation is to incrementally estimate cleaner posterior label distribution. However, there is a possible undesired solution to this approach so that all labels are attained to a single class and base network predicting constant class, which would result in delusional top training accuracy. Therefore additional regularizers are commonly used to make label posterior distribution even. A joint optimization framework for both training base classifier and propagating noisy labels to cleaner labels is presented in [63]. Using expectation-maximization, both classifier parameters and label posterior distribution is estimated in order to minimize the loss. A similar approach is used in [64] with additional compatibility loss conditioned on label posterior. Considering noisy labels are in the minority, this term assures posterior label distribution is not diverged too much from the given noisy label distribution, so that majority of the clean label contribution is not lost. [65, 66] deploy a confidence policy where labels are determined by either network output or given noisy labels, depending on the confidence of the model's prediction. Arguing that in case of noisy labels model first learns correctly labeled data and then overfits to noisy data, [67] aims to extract the probability of a sample being noisy or not from its loss value. To achieve this, the loss of each sample is fitted by a beta mixture model, which models the label noise in an unsupervised manner. [68] proposes a two-level approach. In the first stage, with any chosen inference algorithm, the ground truth labels are determined, and data is divided into two subsets as noisy and clean. In the second stage, an ensemble of weak classifiers is trained on clean data to predict true labels of noisy data. Afterward, these two subsets of data are merged to create the final enhanced dataset. [69] constructs prototypes that are able to represent deep feature distribution of the corresponding class, for each class. Then corrected labels are found by checking similarity among data samples and prototypes. [70] introduces a new parameter, namely *quality embedding*, which represents the trustworthiness of data. Depending on two latent variables, true class probability and quality embedding, an additional network tries to extract the true class of each instance. In a multi-labeled dataset, where each instance has multiple labels representing its content, some labels may be partially missing resulting in partial labels. In the case of partial labels, [71] uses one network to find and estimate easy missing labels and other network to be trained on this corrected data. [158] formulates video anomaly detection as a classification with label noise problem and trains a graph convolutional label noise cleaning network depending on features and temporal consistency of video snippets.

3.1.3 Dataset Pruning

Instead of correcting noisy labels to their true classes, an alternative approach is to remove them. While this would result in loss of information, preventing the harmful impact of noise may result in better performance. In these methods, there is a risk of removing too many samples. Therefore, it is crucial to remove as few samples as possible to prevent unnecessary data loss.

There are two alternative ways for data pruning. First option is to remove noisy samples completely and train the classifier on the pruned dataset. Second option is to just remove the labels of noisy data and transform dataset into two subsets as; labeled and unlabeled data. Then semi-supervised learning algorithms can be employed on the resultant dataset.

3.1.3.1 Removing Data

The most straightforward approach is to remove instances that are misclassified by the base network [72]. [73] uses an ensemble of filtering methods, where each of them assigns a noisiness level for each instance. Then, these predictions are combined, and data with the highest noisiness level predictions are removed. [74] extends this work with label correction. If the majority of noise filters predict the same label for the noisy instance, it's label is corrected to the predicted label. Otherwise, it is removed from the dataset. In [75], with the help of a probabilistic classifier, training data divided into two subsets: confidently clean and noisy. Noise rates are estimated according to the sizes of these subsets. Finally, relying on output confidence of base network on data instances, the number of most unconfident samples is removed according to the estimated noise rate. In [76], transfer learning is used, so that network trained on a clean dataset from a similar domain is fine-tuned on the noisy dataset for relabeling. Afterward, the network is again trained on relabeled data to re-sample the dataset to construct a final clean dataset. In [77], the learning rate is adjusted cyclicly to change network status between underfitting and overfitting. Since, while underfitted, noisy samples cause high loss, samples with large noise during this cyclic process are removed. [78] first train network on noisy data and extract feature vectors by using this model. Afterward, data is clustered with K-means algorithm running on extracted features, and outliers are removed. [159] provides a comparison of performances of various noise-filtering methods for crowd-sourced datasets.

3.1.3.2 Removing Labels

The simplest option is to employ straightforward semi-supervised training on labeled and unlabeled data [79]. Alternatively, label removing can be done iteratively in each epoch to dynamically update dataset for better utilization of semi-supervised learning. [80] uses consistency among given label and moving average of model predictions to evaluate if the given label is noisy or not. Then model is trained on clean samples on the next iteration. This procedure continues until convergence to the best estimator. Same approach is used in [81] with a little tweak. Instead of comparing with given labels, moving average of predictions are compared with predicted labels in the current epoch. In order to avoid the data selection biased caused by one model, [1] uses two models to select unlabeled set for each other. Afterward, each network is trained in semi-supervised learning manner on the dataset selected by its peer network. Another approach in this class is to train a network on labeled and unlabeled data, and then use it to relabel noisy data [82]. Assuming that correctly labeled data account for majority, [83] proposes to randomly split dataset to labeled and unlabeled subgroups. Then, labels are propagated to unlabeled data using similarity index among instances. This procedure repeated to produce multiple labels per instance and then final label is set with majority voting.

3.1.4 Sample Choosing

A widely used approach to overcome label noise is to manipulate the input stream to the classifier. Guiding the network with choosing the right instances to feed can help classifier finding its way easier in the presence of noisy labels. It can be formulated as follows

$$\hat{R}_{l,\mathcal{D}}(f) = \frac{1}{N}\sum_{i=1}^{N} V(x_i, y_i) l(f_\theta(x_i), \tilde{y_i})) \tag{3.3}$$

where $V(x_i, y_i) \in \{0, 1\}$ is a binary operator that decides to whether use the given data (x_i, y_i) or not. If $V(x_i, y_i) = 1$ for all data, then it turns out to be classical risk minimization (Equation 3.3). If V happens to be a static function, which means choosing the same samples during whole training according to a predefined rule, then it turns out to be dataset pruning, as explained in subsection 3.1.3. Differently, sample choosing methods continuously monitor the base classifier and select samples to be trained on for the next training iteration. The task can be seen as drawing a path through data that would mimic the noise-free distribution of $\mathcal{D}$. Since these methods operate outside of the existing system, they are easier to attach to the existing algorithm at hand by just manipulating the input stream. However, it is vital to keep the balance so that system does not ignore unnecessarily large quantities of data. Additionally, these methods prioritize low loss samples, which results in a slow learning rate since hard informative samples are considered only in the later stages of training. Two major approaches under this group are discussed in the following subsections.

3.1.4.1 Curriculum Learning

Curriculum learning (CL) [160], inspired from human cognition, proposes to start from easy samples and go through harder samples to guide training. This is also called *self-paced learning* [161, 162], when prior to sample hardness is not known and inferred from the loss of the current model on that sample. In noisy label framework, clean labeled data can be accepted as easy task while noisily labeled data is harder task. Therefore, the idea of CL can be transferred to label noise setup as starting from confidently clean instances and go through noisier samples as the classifier gets better. Various screening loss functions are proposed in [84] to sort instances according to their noisiness level. Teacher-student approach is implemented in [85], where the task of the teacher is to choose confidently clean samples for the student. Instead of using a predefined curriculum, the teacher constantly updates its curriculum depending on the outputs from the student. Arguing that CL slows down the learning speed, since it focuses on easy samples, [86] suggests choosing uncertain samples that are predicted incorrectly sometimes and correctly on others during training. These samples assumed to be probably not noisy since noisy samples should be predicted incorrectly all the time. Arguing that it is hard to optimize 0-1 loss, *curriculum loss* that chooses samples with low loss values for loss calculation, is proposed as an upper bound for 0-1 loss in [87]. In [88], data is split into subgroups according to their complexities. Since less complex data groups expected to have more clean labels, training will start from less complex data and go through more complex instances as the network gets better. Next samples to be trained on can be chosen by checking the consistency of the label with the network prediction. In [89], if both label and model prediction of the given sample is consistent, it is used in the training set. Otherwise, the model has a right to disagree. Iteratively this provides better training data and better model. However, there is a risk of the model being too skeptical and choosing labels in a delusional way; therefore, consistency balance should be established.

3.1.4.2 Multiple Classifiers

Some works use multiple classifiers to help each other to choose the next batch of data to train on. This is different than the teacher-student approach since none of the networks is supervising the other but they rather help each other out. This can provide robustness since networks can correct each other's mistakes due to their differences in learned representations. For this setup to work, the initialization of the classifiers is important. They are most likely to be initialized with a different subset of the data. If they have the same weight initializations, then there happens no update since they will both agree to disagree with labels. In [90] label is assumed to be noisy if both networks disagree with the given label, and update on model weights happens only when the prediction of two networks conflicts. The paradigm of *co-teaching* is introduced in [91], where two networks select the next batch of data for each other. The next batch is chosen as the data batch, which has small loss values according to pair network. It is claimed that using one network accumulates the noise-related error, whereas two networks filter noise error more successfully. The idea of co-teaching is further improved by iterating over data where two networks disagree, to prevent two networks converging each other with increasing number of epochs [92, 93]. Another work using co-teaching first trains two networks on a selected subset for a given number of epochs and then moves to the full dataset [94].

3.1.5 Sample Importance Weighting

Similar to sample choosing, training can be made more effective by assigning weights to instances according to their estimated noisiness level. This has an effect of emphasizing cleaner instances for better update on model weights. Following empirical risk is minimized by these algorithms

$$\hat{R}_{l,\mathcal{D}}(f) = \frac{1}{N}\sum_{i=1}^{N}\beta(x_i, y_i)l(f_\theta(x_i), \tilde{y}_i)) \tag{3.4}$$

where $\beta(x_i, y_i)$ determines the instance dependent weight. If β would be binary, then formulation is the same with sample choosing, as explained in subsection 3.1.4. Differently, here β is not binary and has a different value for each instance. Just like in sample choosing algorithms, β is a dynamic function, which means weights for instances keep changing during the training. Therefore, it is commonly a challenge to prevent β changing too rapidly and sharply, such that it disrupts the stabilized training loop. Moreover, these methods commonly suffer from accumulated errors so that they can easily get biased towards a certain subset of data. There are various methods proposed to obtain optimal β to fade away the negative effects of noise.

The simplest approach would be, in case of availability of both clean and noisy data, weighting clean data more [53]. However, this utilizes information poorly; moreover, clean data is not always available. Works of [95] and [96], uses meta-learning paradigm to determine the weighting factor. In each iteration, gradient descent step on given mini-batch for weighting factor is performed, so that it minimizes the loss on clean validation data. A similar method is adopted in [97], but instead of implicit calculation of the weighting factor, multi layer perceptron (MLP) is used to estimate the weighting function. *Open-set noisy labels*, where data samples associated with noisy labels might belong to a true class that is not present in the training data, are considered in [98]. Siamese network is trained to detect noisy labels by learning discriminative features to apart clean and noisy data. Noisy samples are iteratively detected and pulled from clean samples. Then, each iteration weighting factor is recalculated for noisy samples, and the base classifier is trained on whole dataset. [26] also iteratively separates noisy samples and clean samples. On top of that, not to miss valuable information from clean hard samples, noisy data are weighted according to their noisiness level, which is estimated by pLOF [163]. [99] introduces *abstention*, which gives option to abstain samples, depending on their loss value, with an abstention penalty. Therefore, the network learns to abstain from confusing samples, and with the abstention penalty, the tendency to abstain can be adjusted. In [100], weighting factor is conditioned on distribution of training data, $\beta(X, Y) = P_{\mathcal{D}}(X, Y)/P_{\mathcal{D}_n}(X, \tilde{Y})$. The same methodology is extended to the multi-class case in [101]. In [102], the weighting factor is determined by checking instance similarity to its representative class prototype in the feature domain. [103] formulates the problem as transfer learning where the source domain is noisy data and target domain is a clean subset of data. Then weighting in source domain is arranged in a way to minimize target domain loss. [104] uses θ values of samples in θ-distribution to calculate their probability of being clean and use this information to weight clean samples more in training.

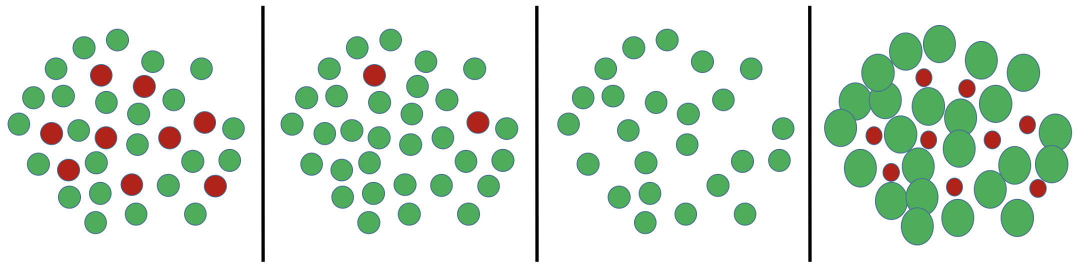


Figure 3.2: Illustration of different types of algorithms. Starting from left; 1) representation of samples from a single class in the 2D space. Green samples represent the clean samples and red ones represent noisy samples. 2) label noise cleaning algorithms aim to correct the labels of noisy data 3) dataset pruning methods aim to eliminate noisy data (or just their labels) 4) sample importance weighting algorithms aim to up-weight clean samples and down-weight noisy samples (which is illustrated by size)

3.1.6 Labeler Quality Assessment

As explained in section 2.3, there can be several reasons for dataset to be labeled by multiple annotators. Each labeler may have different level of expertise and their labels may occasionally contradict with each other. This is a common case in crowd-sourced data [164, 165, 166] or datasets which requires high level of expertise such as medical imaging [14]. Therefore, modeling and using labeler characteristic can significantly increase the performance [23].

In this setup there are two unknowns namely; noisy labeler characteristic and ground truth labels. One can estimate both with expectation-maximization [105, 106, 24]. If noise is assumed to be y-dependent, labeler characteristic can be modeled with noise transition matrix, just like in subsection 3.1.1. [107] adds a regularizer to the loss function, which is the sum of traces of annotator confusion matrices, in order to force sparsity on each labeler's estimated confusion matrix. Similar approach is implemented in [108], where crowd-layer is added to the end of network. In [109], xy-dependent noise is also considered by taking image complexities into account as well. Human annotators and computer vision system are used mutually in [110], where consistency among predictions of these two components are used to evaluate the reliability of labelers. [111] deals with the noise when labeler omits a tag in the image. Therefore, instead of noise transition matrix for labelers, omitting probability variable is used, which is estimated together with true class using expectation-maximization algorithm. Separate softmax layers are trained for each annotator in [23] and an additional network to predict the true class of data depending on the outputs of labeler specific networks and features of data. This setup enables to model each labeler and their overall noise structure in separate networks.

3.1.7 Discussion

Visual illustration of some of the methods are presented in Figure 3.2. Noise model based methods are heavily dependent on the accurate estimate of the noise structure. This brings a dilemma. For better noise model one needs better estimators, and for better estimators it is necessary to have a better estimate of underlying noise. Therefore, many approaches can be seen as an expectation-maximization of both noise estimation and classification. However, it is essential to prevent the system diverging from reality, therefore regularizing noise estimates and not letting it getting delusional is important. In order to achieve this, works in literature commonly make assumptions about the underlying noise structure, which damages their applicability to different setups. On the other hand, this lets any prior information about the noise to be inserted to the system for an head-start. It is also useful to handle domain-specific noise. One another advantage of these algorithms is they decouple noise estimation and classification tasks. Therefore, they are easier to implement on an existing classification algorithm at hand.

3.2 Noise Model Free Methods

These methods aim to achieve label noise robustness without explicitly modeling it, but rather designing robustness in the proposed algorithm. Noisy data is treated as anomaly and therefore these methods are in similar line with overfit avoidance. They commonly rely on internal noise tolerance of the classifier and aim to boost performance by regularizing undesired memorization of noisy data. Various methodologies are presented in the following subsections.

3.2.1 Robust Losses

A loss function is said to be noise robust if the classifier learned with both noisy and noise-free data, achieves the same classification accuracy [112]. Algorithms under this section aims to design loss function in such a way that the existence of the noise would not decrease the performance. However, it is shown that noise can badly affect the performance even for the robust loss functions [12]. Moreover, these methods treat both noisy and clean data in the same way, which prevents the utilization of any prior information over data distribution.

In [112], it is shown that certain non-convex loss functions, such as 0-1 loss, has noise tolerance much more than commonly used convex losses. Extending this work [113, 114] derives sufficient conditions for a loss function to be noise tolerant for uniform noise. Their work shows that, if the given loss function satisfies $\sum_k l(f_\theta(x), k) = C, \forall x \in X$ where C is a constant value, then loss function is tolerant to uniform noise. In this content, they empirically show that none of the standard convex loss functions has noise robustness while 0-1 loss has, up to a certain noise ratio. However, 0-1 loss is non-convex and non-differentiable; therefore, surrogate loss of 0-1 loss is proposed in [115], which is still noise sensitive. Widely used *categorical cross entropy (CCE)* loss is compared with *mean absolute value of error (MAE)* in the work of [116], where it is shown empirically that mean absolute value of error is more noise tolerant. [117] shows that the robustness of MEA is due to its weighting scheme. While CCE is sensitive to abnormal samples and produces bigger gradients in magnitude, MAE treats all data points equally, which would result in an underfitting of data. Therefore, *Improved mean absolute value of error (IMAE)*, which is an improved version of MAE, is proposed in [117], where gradients are scaled with a hyper-parameter to adjusts weighting variance of MAE. [118] also argues that MAE provides a much smaller learning rate than CCE; therefore, a new loss function is suggested, which combines the robustness of MAE and implicit weighting of CCE. With a tuning parameter, characteristic of the loss function can be adjusted in a line from MAE to CCE. Loss functions are commonly not symmetric, meaning that $l(f_\theta(x_i), y_i) \neq l(y_i, f_\theta(x_i))$. Inspired from the idea of symmetric KL-divergence, [119] proposes symmetric cross entropy loss $l_{SCE}(f_\theta(x_i), y_i) = l(f_\theta(x_i), y_i) + l(y_i, f_\theta(x_i))$ to battle noisy labels.

Given that noise prior is known, [120] provides two surrogate loss functions using the prior information about label noise, namely, unbiased and weighted estimator of the loss function. [121] considers asymmetric omission noise for the binary classification case, where the task is to find road pixels from a satellite map image. Omission noise makes the network less confident about its predictions, so they modified cross-entropy loss to penalize network less for making wrong but confident predictions since these labels are more likely to be noisy. Instead of using distance-based loss, [122] proposes to use information-theoretic loss, in which determinant based mutual information [167] between given labels and predictions are evaluated for loss calculation. Weakly supervised learning with noisy labels are considered in [123], and necessary conditions for loss to be noise tolerant are drawn. [124] shows that classification-calibrated loss functions are asymptotically robust to symmetric label noise. Stochastic gradient descent with robust losses are analyzed in general [125] and shown to be more robust to label noise than its counterparts.

3.2.2 Meta Learning

As mentioned in section 2.4, recent works show that meta-learning achieves promising results in various fields. Designing a task beyond classical supervised learning in meta learning fashion has been used to deal with label noise as well. A meta task is defined as predicting the most suitable method, among family of methods, for a given noisy dataset in [126]. *Pumpout* [127] presents a meta objective as recovering the damage done by noisy samples by erasing their effect on model via *scaled gradient ascent*. As a meta learning paradigm, model-agnostic-meta-learning (MAML) [44] seeks for optimal weight initialization that can easily be fine-tuned for a desired objective. A similar mentality is used in [128] for noisy labels, which aims to find noise-tolerant model parameters that are less prone to noise under teacher-student training framework [168, 169]. Multiple student networks are fed with data corrupted by synthetic noise, and meta objective is defined to maximize consistency with teacher outputs, which are obtained from raw data without synthetic noise. Therefore, student networks are forced to find most noise robust weight initialization such that weight update will still be consistent after training an epoch on synthetically corrupted data. Then, final classifier weights are set as an exponential moving average of student networks.

Alternatively, in the case of available clean data, a meta objective can be defined to utilize this information. The approach used in [129] is to train a teacher network in a clean dataset and transfer its knowledge to student network for the purpose of guiding training process in the presence of mislabeled data. They used *distillation* technique proposed in [45] for controlled transfer of knowledge from teacher to student. In [130, 131] the target network is trained on excessive noisy data, and the confidence network is trained on clean subset. Inspiring from [43], the confidence network's task is to control the magnitude of gradient updates to the target network so that noisy labels are not resulting in updating gradients. [132] uses clean data to produce soft labels for noisy data, for which the classifier trained on it would give the best performance on the clean data. As a result, it seeks for optimal label distribution to provide most noise robust learning for the base classifier.

3.2.3 Regularizers

Regularizers are well known to prevent DNNs from overfitting noisy labels. From this perspective, these methods treat performance degradation due to noisy data as overfitting to noise. Even though this assumption is mostly valid in random noise, it may not be the case for more complex noises. Some widely used techniques are weight decay, dropout [133], adversarial training [134], mixup [135], label smoothing [136, 137]. [138] shows that pre-training has a regularization effect in the presence of noisy labels. In [139] an additional softmax layer is added, and dropout regularization is applied to this layer, arguing that it provides more robust training and prevents memorizing noise due to randomness of dropout [133]. [140] proposes a complexity measure to understand if the network starts to overfit. It is shown that learning consists of two steps: 1) dimensionality compression, that models low-dimensional subspaces which closely match the underlying data distribution, 2) dimensionality expansion, that steadily increases subspace dimensionality in order to overfit the data. The key is to stop before the second step. *Local intrinsic dimensionality* [170] is used to measure complexity of trained model and stop before it starts to overfit. [141] takes a pre-trained network on a different domain and fine-tunes it for the noisy labeled dataset. Groups of image features are formed, and group sparsity regularization is imposed so that model is forced to choose relative features and up-weights the reliable images.

3.2.4 Ensemble Methods

It is well known that bagging is more robust to label noise than boosting [171]. Boosting algorithms like AdaBoost puts too much weight on noisy samples, resulting in overfitting the noise. However, the degree of label noise robustness changes for the chosen boosting algorithm. For example, it is shown that BrownBoost and Logit-Boost are more robust than AdaBoost [142]. Therefore, noise-robust alternatives of AdaBoost is proposed in literature, such as noise detection based AdaBoost [143], rBoost [144], RBoost1&RBoost2 [145] and robust multi-class AdaBoost [146].

3.2.5 Others

Complementary labels define classes that observations do not belong to. For example, in the case of ten classes, there is one true class for an instance and nine complementary classes. Since annotators are less likely to mislabel, some works propose to work in complementary label space [147, 148]. [149] uses reconstruction error of autoencoder to discriminate noisy data from clean data, arguing that noisy data tend to have bigger reconstruction error. In [150], a special setup is considered where dataset consists of noisy and *less-noisy* data for binary classification task. [151] aims to extract the quality of data instances. Assuming that the training dataset is generated from a mixture of target distribution and other unknown distributions, it estimates the quality of data samples by checking the consistency between generated and target distributions.

Prototype learning aims to construct prototypes, that can represent features of a class, in order to learn clean representations. Some works in the literature [152, 153] propose to create clean representative prototypes for noisy data, so that base classifier can be trained on them instead of noisy labels.

In multiple-instance learning, data are grouped in clusters, called bags, and each bag is labeled as positive if there is at least one positive instance in it and negative otherwise. The network is fed with a group of data and produces a single prediction for each bag by learning the inner discriminative representation of data. Since the group of images is used and one prediction is produced, existence of noisy labels along with true labels in a bag has less impact on learning. In [154], authors propose to effectively choose training samples from each bag by minimizing the total bag level loss. Extra model is trained in [155] as attention model, which chooses parts of the images to be focused on. Aim is to focus on few regions on correctly labeled image and not focus on any region for mislabeled images.

3.2.6 Discussion

Methods belonging to this category, in overall, treat noisy data as an anomaly. Therefore, they are in a similar line with overfit avoidance and anomaly detection. Even though this assumption may be quite valid for random noise, it loses its validity in case of more complicated and structured noises. Since noise modeling is not decoupled from classification task explicitly, proposed methods are, in the general sense, embedded into the existing algorithm. This prevents their quick deployment to the existing system at hand. Moreover, algorithms belonging to meta-learning and ensemble methods can be computationally costly since they require multiple iterations of training loops.

CHAPTER 4

SYNTHETIC LABEL NOISE GENERATION

As presented in Section 2.2, there are various label noise types. Generating random noise and class-dependent noise is pretty straightforward. On the other hand, there are multiple ways of producing feature-dependent noisy labels. It consists of two steps: 1) mapping instances to feature domain and picking ambiguous samples, 2) flipping label to most probable class. Different implementations of these steps result in different types of feature-dependent noises.

This chapter consists of two parts. The first part analyses the various implementations of label noise generation algorithms from the literature. Then, a novel feature-dependent label noise generation algorithm is proposed in the second part.

4.1 Methods from the Literature

This section presents the methodologies to produce different types of synthetic label noise. *Uniform noise* and *class-dependent noise* can be represented with noise transition matrix N where N_{ij} represents the probability of flipping label from class i to j. Since noise transition matrix consists of probabilities, $\sum_j N_{ij} = 1$. On the other hand, in *feature-dependent noise*, each instance has its own transition probability depending on its features. Therefore, it can not be generated using a noise transition matrix. The following sections will describe the process of generating these types of noises. Different noise types are visualized with T-SNE plots in Figure 2.1.

4.1.1 Uniform Noise

Flipping the probability of a label from its true class to any other class is equally distributed. Many works in literature use synthetic uniform label noise by just flipping labels randomly for a given percentage of data instances [140, 96, 139, 59].

For this type of noise, each entry in the noise transition matrix, besides diagonal ones, are equally distributed. The noise transition matrix can be defined as follows.

$$N_{ij} = \begin{cases} p & \text{if } i = j \\ \dfrac{1-p}{C-1} & \text{if } i! = j \end{cases} \tag{4.1}$$

4.1.2 Class-Dependent Noise

The flipping probability of the label depends on the true class of the data instance. This is mostly represented by a confusion matrix and can be designed in different ways. The easiest way is to attain inter-class transition probabilities just random [103] so that there is still class dependence since transition probabilities are given according to classes but without any correlation to class similarities. In a more structured way, noise transition matrix can be designed in a way that similar classes have a bigger probability to be flipped to each other [48, 128, 64, 54, 49]. Some works use pairwise noise, in which transition from one class can only be defined to one another class [91, 95, 127, 92]. Work of [122] checks the popularity of classes and constructs transition matrix so that mislabeling happens from popular class to unpopular class or vice versa.

4.1.3 Feature-Dependent Noise

The probability of mislabeling depends on features of instances. In order to generate feature-dependent noise, features of each instance should be extracted, and their similarities to other instances from different classes should be evaluated. Unlike uniform and class-dependent noise, there are much fewer implementations of synthetic feature-dependent label noise. One particular work in this field is [172], where data is clustered with the kNN algorithm, and labels are flipped randomly for clusters of data. This method provides concentrated noise in the feature space. This type of synthetic noise does not evaluate the instance similarities and, therefore, differs from our proposed approach. Alternatively, in case there is a surrounding text for each image in the dataset, some works create noisy labels from the interpretations of these texts [24, 129, 57], assuming surrounding texts are related to features of data. However, this approach is restricted to datasets with surrounding user-defined texts, which is not the case for most of the time.

4.2 Proposed Noise Generation Algorithm

It should be noted that toy datasets with synthetic label noise are only used for the development of the noise-robust algorithm. The main goal of this thesis is to develop a noise robust algorithm for real-world applications. Therefore, generated artificial noise should be as close to real-world scenarios as possible. Obviously, the most realistic label noise is the feature-dependent noise. Annotators are highly affected by the attributes of the data and tend to mislabel ambiguous samples. Therefore, the proposed algorithm should manage to achieve two critical stages; finding ambiguous samples and figuring out the most probable noisy label.

Compared to uniform noise and class-dependent noise, feature dependent noise is harder to implement since all samples should be vectorized in the feature domain, and similarities among samples should be calculated. The most similar work on this topic is [172], where labels are flipped for clusters of instances that result in locally-concentrated label noise. However, this approach doesn't utilize the similarities among instances; therefore, it does not fit this work's requirements.

Once mapped to the feature domain, picking ambiguous samples is a straightforward process. One can identify the samples that are close to decision boundaries and flip their labels to counterpart class. However, finding the mapping function is challenging. One option is to train a deep network on the dataset first and then use it as the feature extractor. However, since the network extracts the features of data that it is trained on, it is prone to overfitting. Since we are especially interested in similarities among instances in feature space, it is desired that samples are sparsely distributed. On the contrary, in the case of overfitting, samples are gathered in a small region in feature space.

Therefore, in this work the idea of *knowledge distillation* [45] is used. In the original work, the authors use distillation to transfer knowledge from the big teacher network to a much smaller student network without decreasing the performance. The idea is mainly motivated by learning from soft labels where the similarity of each instance to each class is emphasized by *temperature* hyperparameter.

Class probabilities on softmax output, beyond the true class probability, are usually very low. But, compared with each other, some classes may have a much higher probability than others, and this carries important information about that data instance, which is also called as *dark knowledge*. By making probability distribution smoother, this relation is emphasized, as shown in Equation 4.2.

$$q_i = \frac{exp(z_i/T)}{\sum_j exp(z_j/T)} \tag{4.2}$$

Where z_i is logit value for class i and q_i is converted probability value for class i. When temperature $T = 1$, formulation turns out to be normal softmax function. Instead of being trained on hard labels, the student network is trained on the weighted sum of hard labels and soft labels produced by the teacher network. So, the loss function is defined as follows,

$$L(y_i, f(x_i)) = \alpha l(y_i, f(x_i)) + (1 - \alpha) l(q_i, f(x_i)) \tag{4.3}$$

where q_i represents the soft labels produced by the teacher network using temperature T and y_i represents the given label.

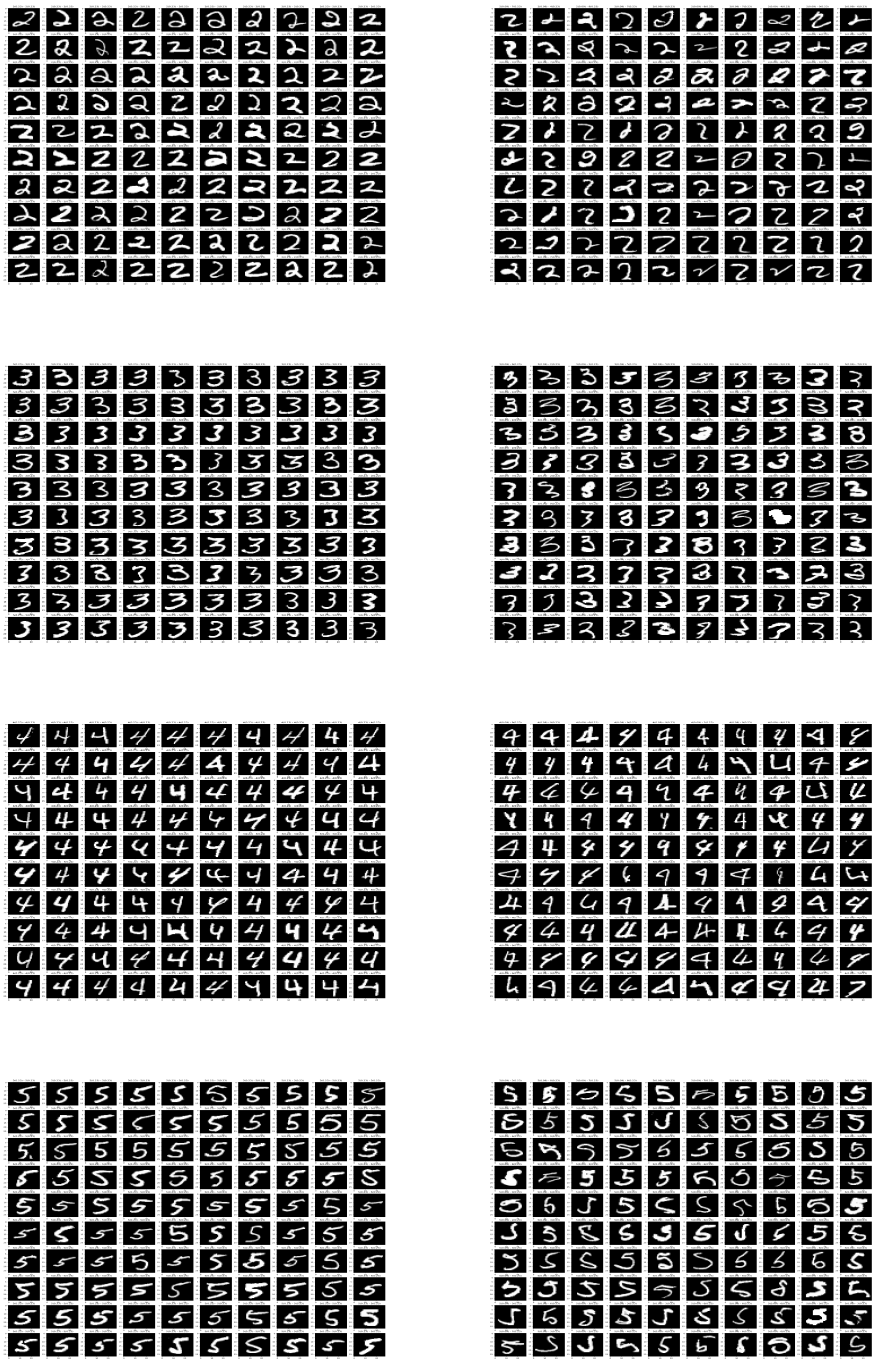
Within the context of this thesis, there is no interest in compressing the network to a smaller network. However, the idea of learning by emphasizing instance similarities can be used to find instances that have similar features with other classes. For that purpose, the student network is at the same size as the teacher network. Firstly, the teacher network is trained on the dataset. Secondly, soft labels are produced from the softmax output of the teacher for a given temperature T. Thirdly, the student network is trained on soft labels and hard labels with a weighting factor α in the loss function. Finally, by checking softmax probabilities of instances, samples that have similar features to other classes are detected, and their labels are flipped to corresponding classes.

To see if the proposed method results in a more sparse distribution of data in feature space, we can check the variance of instances belonging to classes and average over each class as follows

$$\sigma = \frac{\sum_i^N var(Q_i)}{C} \tag{4.4}$$

where Q_i is the feature matrix of instances belonging to class i. Features are extracted from the layer output before the softmax layer. For straightforward training on the MNIST dataset, the network manages to get 99% test accuracy while having $\sigma = 78.3$. On the other hand, the network trained with distillation achieves 95% accuracy while having $\sigma = 563.5$. Its accuracy is comparable to the original network, but learned representations are much more sparse, which is useful to extract similarities.

Figure 4.1 shows the data samples picked by the proposed label noise generation algorithm. Additionally, most reliable data samples are visualized as well. It can be seen that the proposed algorithm successfully manages to pick ambiguous samples.



(a) Confident data samples (b) Ambiguous data samples

Figure 4.1: Data samples chosen by presented synthetic label noise generation algorithm. Left column shows the instances that are confidently classified, so their labels are not changed. On the other hand, right column shows ambiguous samples, for which labels are changed.

CHAPTER 5

META SOFT LABEL GENERATION

Conventional training with stochastic gradient descent fails when trained on noisy data.Various approaches deal with this problem of learning from noisy data by designing different learning frameworks instead of straightforward stochastic gradient descent algorithm. Alternatively, this problem of learning from noisy data can itself be seen as another optimization problem. The sub-optimality of straightforward learning originates from the sub-optimality of labels due to noise. Therefore, by optimizing noisy data labels, the base learning algorithm can be optimized. In this chapter, meta-learning based label noise robust learning algorithm is proposed, which is called **M**eta **S**oft **L**abel **G**eneration (MSLG). The proposed algorithm generates soft-labels for each instance so that the base model is trained with these soft-labels instead of given noisy labels. Soft-labels are determined by meta-objective, which is to minimize the loss of meta-data. Two iterations of training is used in each loop: 1) Update soft-label generation algorithm according to the meta-objective, 2) Update base classifier parameters by newly generated soft-labels. As a result, the *proposed meta objective seeks for soft labels such that update on classification loss would lead network parameters in the direction of minimizing meta loss*. The proposed algorithm differs from conventional label noise cleansing methods in a way that it is not searching for clean hard-labels but rather searching for optimal labels in soft-label space to minimize meta objective. Therefore, propagated soft labels are not necessarily clean labels, but corresponding labels in a different label space. The algorithm proposed in this chapter is previously published in [132], and the content is highly correlated with the original paper.

5.1 Training

The full pipeline of the proposed framework is illustrated in Figure 5.1. Overall, training consists of two phases. Phase-1 is the warm-up training, which aims to provide a stable initial point for phase-2 to start on. After the first phase, the main algorithm is employed in the second phase that concludes the training. Each of the training phases and their iteration steps are explained in the following subsection.

5.1.1 Training Phase-1

It is commonly accepted that, in the presence of noise, deep neural network firstly learn useful representations and then overfit the noise [10]. Therefore, before employing the proposed algorithm, a warm-up training for the base classifier on noisy training data with conventional cross-entropy loss is employed. At this stage, the useful information from the data is leveraged. This is also beneficial for the meta-training stage. Since gradients are taken on the feedback coming from the base classifier, without any pre-training, random feedbacks coming from the base network would cause meta-training to lead in the wrong direction.

5.1.2 Training Phase-2

This phase is the main training stage of the proposed framework. There are three iteration steps, which are executed on each batch of data consecutively. The first two iteration steps are the meta-training step, in which soft-labels are updated. Afterward, in the third iteration step, base classifier parameters are updated by using the updated soft-labels. These three steps are executed for each batch of data consecutively. The following subsections explain training steps in detail.

5.1.2.1 Meta Training Step

The meta training step aims to optimize the conventional training step by updating soft-labels. Therefore, it needs feedback from the conventional training step. For that reason, firstly, a dummy conventional training step is conducted on the base classifier with generated soft-labels. Secondly, meta-loss is calculated using the updated classifier and meta-data. Then, meta-loss is backpropagated for soft-labels $\hat{y}$.

Firstly, posterior model parameters are calculated by taking a stochastic gradient descent step on the classification loss.

$$\hat{\theta} = \theta^{(t)} - \alpha \nabla_{\theta} \mathcal{L}_c(f_{\theta}(x), \hat{y}^{(t)}) \Big|_{\theta^{(t)}} \text{, and} \tag{5.1}$$

$$\mathcal{L}_c(f_{\theta}(x), \hat{y}^{(t)}) = \frac{1}{N_b} \sum_{i=1}^{N_b} l_c(f_{\theta}(x_i), \hat{y}_i^{(t)}) \tag{5.2}$$

where $x_i \in \mathcal{D}_n$ and $\hat{y}_i^{(t)}$ is the corresponding predicted label at time step t. Inspired from [64], classification loss l_c is set as KL-divergence loss as follows

$$l_c = KL(f_{\theta}(x_i) || \hat{y}_i) \tag{5.3}$$

Justification for the choice of loss function is provided in section 5.3. Afterward, meta-loss is calculated with the feedback coming from updated parameters.

$$\mathcal{L}_{meta}(f_{\hat{\theta}}(x_m), y_m) = \frac{1}{N_b} \sum_{j=1}^{N_b} l_{cce}(f_{\hat{\theta}}(x_{m,j}), y_{m,j}) \tag{5.4}$$

where $x_{m,j}, y_{m,j} \in \mathcal{D}_m$ and l_{cce} represents the conventional categorical cross-entropy loss. Finally, label predictions are updated by minimizing the meta loss $\mathcal{L}_{meta}$.

$$\hat{y}^{(t+1)} = \hat{y}^{(t)} - \beta \nabla_{\hat{y}} \mathcal{L}_{meta}(f_{\hat{\theta}}(x_m), y_m) \Big|_{\hat{y}^{(t)}} \tag{5.5}$$

Mathematical reasoning for the meta-update is further presented in Section 5.4.

5.1.2.2 Conventional Training Step

In this phase, the base network is trained on two losses. The first loss is classification loss, which ensures network predictions are consistent with estimated soft labels.

$$\mathcal{L}_c(f_\theta(x), \hat{y}^{(t+1)}) = \frac{1}{N_b}\sum_{i=1}^{N_b} l_c(f_\theta(x_i), \hat{y}_i^{(t+1)}) \tag{5.6}$$

Notice that this is the same loss formulation with Equation 5.1, but with updated soft labels $\hat{y}_i^{(t+1)}$.

Moreover, Inspired from [63], entropy loss is defined as follows

$$\mathcal{L}_e(f_\theta(x)) = -\frac{1}{N_b}\sum_{i=1}^{N_b}\sum_{j=1}^{C} f_\theta^j(x_i)\log(f_\theta^j(x_i)) \tag{5.7}$$

Entropy loss forces network predictions to peak at only one class. Since the base classifier is trained on predicted soft labels, which can peak at multiple locations, this is useful to prevent training loop to saturate. Finally, base classifier parameters are updated with stochastic gradient descent on these two losses.

$$\theta^{(t+1)} = \theta^{(t)} - \lambda\nabla_\theta \left(\mathcal{L}_c(f_\theta(x), \hat{y}^{(t+1)}) + \mathcal{L}_e(f_\theta(x))\right)\Big|_{\theta^{(t)}} \tag{5.8}$$

Notice that there are three different learning rates α, β and λ for each step. The overall algorithm is summarized in algorithm 2.

5.2 Formulation of $\hat{y}$

In MSLG, label distribution y^d is maintained for all training samples x_i. Following [64], y^d is initialized using noisy labels $\tilde{y}$ with the following formula

$$y^d = K\tilde{y} \tag{5.9}$$

where K is a large constant. Then softmax is applied to get normalized soft labels

$$\hat{y} = softmax(y^d) \tag{5.10}$$

Algorithm 2: Meta Soft Label Generation

```
Input: Training data $\mathcal{D}_n$, meta-data $\mathcal{D}_m$, batch size $N_b$, phase-1 epoch
       number $e_{phase1}$, phase-2 epoch number $e_{phase2}$,
Output: Base classifier parameters $\theta$, Generated soft labels $\hat{y}$

$epoch = 0$;
// ----------------- Phase-1 training -----------------
for $epoch < e_{phase1}$ do
    $\theta^{(t+1)} = ConventionalTrain(\theta^{(t)}, \mathcal{D}_n)$    // Warm-up training with noisy data
    $epoch = epoch + 1$
end

// ----------------- Phase-2 training -----------------
$\hat{y} = softmax(K\tilde{y})$    // Initialize soft-labels Equation 5.9&5.10
for $epoch < e_{phase2}$ do
    foreach batch do
        $\{x\} \leftarrow$ GetBatch($\mathcal{D}_n$,$N_b$)
        $\{x_m, y_m\} \leftarrow$ GetBatch($\mathcal{D}_m$,$N_b$)

        // ----------------- Iteration step-1 -----------------
        $\mathcal{L}_c(f_\theta(x), \hat{y}^{(t)}) = \frac{1}{N_b}\sum_{i=1}^{N_b} l_c(f_\theta(x_i), \hat{y}_i^{(t)})$    // Loss with soft-labels by Equation 5.2
        $\hat{\theta} = \theta^{(t)} - \nabla_\theta \mathcal{L}_c(f_\theta(x), \hat{y}^{(t)})$    // Update $\theta$ by Equation 5.1

        // ----------------- Iteration step-2 -----------------
        $\mathcal{L}_{meta}(f_{\hat{\theta}}(x_m), y_m) =$
            $\frac{1}{N_b}\sum_{j=1}^{N_b} l_{cce}(f_{\hat{\theta}}(x_{m,j}), y_{m,j})$    // Loss with $\hat{\theta}$ on meta-data by Equation 5.4
        $\hat{y}^{(t+1)} = \hat{y}^{(t)} - \beta\nabla_{\hat{y}}\mathcal{L}_{meta}(f_{\hat{\theta}}(x_m), y_m)$    // Update $\hat{y}$ by Equation 5.5

        // ----------------- Iteration step-3 -----------------
        $\mathcal{L}_c(f_\theta(x), \hat{y}^{(t+1)}) =$
            $\frac{1}{N_b}\sum_{i=1}^{N_b} l_c(f_\theta(x_i), \hat{y}_i^{(t+1)})$    // Loss with new soft-labels by Equation 5.6
        $\mathcal{L}_e(f_\theta(x)) =$
          $-\frac{1}{N_b}\sum_{i=1}^{N_b}\sum_{j=1}^{C} f_\theta^j(x_i)\log(f_\theta^j(x_i))$    // Entropy-loss by Equation 5.7
        $\theta^{(t+1)} = \theta^{(t)} -$
          $\lambda\nabla_\theta\left(\mathcal{L}_c(f_\theta(x), \hat{y}^{(t+1)}) + \mathcal{L}_e(f_\theta(x))\right)$    // Update $\theta$ by Equation 5.8
    end
    $epoch = epoch + 1$;
end
```

This setup provides unconstrained learning for y^d while producing valid soft labels $\hat{y}$ all the time.

5.3 Reasoning of Classification Loss

Inspired from [64], the meta loss is defined as KL-divergence loss with a slight trick. KL-divergence is formulated as follows:

$$KL(P||Q) = \sum_{x \in X} P(x) \log \left(\frac{P(x)}{Q(x)} \right) \tag{5.11}$$

KL-divergence loss is asymmetric, which means

$$KL(Q||P) \neq KL(P||Q) \tag{5.12}$$

Therefore there are two different configurations. First options is:

$$\begin{gathered} \mathcal{L}_{c,1} = \frac{1}{N_b} \sum_{i=1}^{N_b} KL(\hat{y}_i || f_\theta(x_i)), where \\ KL(\hat{y}_i || f_\theta(x_i)) = \sum_{j=1}^{C} \hat{y}_i^j \log \left(\frac{\hat{y}_i^j}{f_\theta^j(x_i)} \right) \end{gathered} \tag{5.13}$$

which produces the following gradients

$$\frac{\partial \mathcal{L}_{c,1}}{\partial f_\theta^j(x_i)} = -\frac{\hat{y}_i^j}{f_\theta^j(x_i)} \tag{5.14}$$

Second possible configuration of loss function is

$$\begin{gathered} \mathcal{L}_{c,2} = \frac{1}{N_b} \sum_{i=1}^{N_b} KL(f_\theta(x_i) || \hat{y}_i), where \\ KL(f_\theta(x_i) || \hat{y}_i) = \sum_{j=1}^{C} f_\theta^j(x_i) \log \left(\frac{f_\theta^j(x_i)}{\hat{y}_i^j} \right) \end{gathered} \tag{5.15}$$

which produces the following gradients

$$\frac{\partial \mathcal{L}_{c,2}}{\partial f_\theta^j(x_i)} = 1 + \log\left(\frac{f_\theta^j(x_i)}{\hat{y}_i^j}\right) \tag{5.16}$$

Lets assume a classification task where true label is 2 ($y_i^2 = 1$) but noisy label is given as 5 ($\tilde{y}_i^5 = 1$). Now lets consider two updates on $\hat{y}_i^2$ and $\hat{y}_i^5$

- **Case $\hat{\boldsymbol{y}}_{\boldsymbol{i}}^{\boldsymbol{j=2}}$:** $\hat{y}_i^2$ is initially very small, therefore $f_\theta^2(x_i) \gg \hat{y}_i^2$. In that case $\mathcal{L}_{c,1}$ will produce a very small gradients (5.14) while $\mathcal{L}_{c,2}$ will produce a medium positive gradient (5.16) as desired.
- **Case $\hat{\boldsymbol{y}}_{\boldsymbol{i}}^{\boldsymbol{j=5}}$:** $\hat{y}_i^5$ initially has peak value; however, due to internal robustness of network we expect $f_\theta^5(x_i) \ll \hat{y}_i^5$. In that case $\mathcal{L}_{c,1}$ produce large negative gradient (5.14) while $\mathcal{L}_{c,2}$ produce medium negative gradients (5.16).

As a result the following statement can be concluded: $\mathcal{L}_{c,1}$ focuses on learning from y_i^5 (noisy label) while $\mathcal{L}_{c,2}$ focuses on positive learning from y_i^2 (correct label) and negative learning from y_i^5 (noisy label). Therefore, $\mathcal{L}_{c,2}$ is better choice for theta learning objective.

Even though given loss formulation may be seen same as [64], its usage it totally different. While [64] uses $\nabla_{\hat{y}}\mathcal{L}_c$ to update $\hat{y}$, the proposed algorithm uses second order derivative of $\nabla_{f_\theta}\mathcal{L}_c$ as a meta-objective which is further explained in Section 5.4.

5.4 Meta Objective

From Equation 5.5, update term for $\hat{y}$ is as follows

$$\beta \nabla_{\hat{y}} \mathcal{L}_{meta}(f_{\hat{\theta}}(x_m), y_m)\Big|_{\hat{y}^{(t)}} \tag{5.17}$$

$$= \beta \frac{\partial \mathcal{L}_{meta}(f_{\hat{\theta}}(x_m), y_m)}{\partial \hat{\theta}} \frac{\partial \hat{\theta}}{\partial \hat{y}}\Big|_{\hat{y}^{(t)}} \tag{5.18}$$

$$= \beta \frac{\partial \mathcal{L}_{meta}(f_{\hat{\theta}}(x_m), y_m)}{\partial \hat{\theta}} \frac{\partial}{\partial \hat{y}} \left(-\alpha \frac{\partial \mathcal{L}_c(f_\theta(x), \hat{y}^{(t)})}{\partial \theta} \right) \Bigg|_{\hat{y}^{(t)}} \tag{5.19}$$

$$= \beta \frac{1}{N_b} \sum_{j=1}^{N_b} \frac{\partial l_{cce}(f_{\hat{\theta}}(x_{m,j}), y_{m,j})}{\partial \hat{\theta}} \frac{\partial}{\partial \hat{y}} \left(-\frac{\alpha}{N_b} \sum_{i=1}^{N_b} \frac{\partial l_c(f_\theta(x_i), \hat{y}_i^{(t)})}{\partial \theta} \right) \Bigg|_{\hat{y}^{(t)}} \tag{5.20}$$

$$= -\frac{\alpha\beta}{N_b N_b} \frac{\partial}{\partial \hat{y}} \left(\sum_{j=1}^{N_b} \frac{\partial l_{cce}(f_{\hat{\theta}}(x_{m,j}), y_{m,j})}{\partial \hat{\theta}} \sum_{i=1}^{N_b} \frac{\partial l_c(f_\theta(x_i), \hat{y}_i^{(t)})}{\partial \theta} \right) \Bigg|_{\hat{y}^{(t)}} \tag{5.21}$$

$$= -\frac{\alpha\beta}{N_b N_b} \frac{\partial}{\partial \hat{y}} \left(\sum_{i=1}^{N_b} \sum_{j=1}^{N_b} \frac{\partial l_c(f_\theta(x_i), \hat{y}_i^{(t)})}{\partial \theta} \frac{\partial l_{cce}(f_{\hat{\theta}}(x_{m,j}), y_{m,j})}{\partial \hat{\theta}} \right) \Bigg|_{\hat{y}^{(t)}} \tag{5.22}$$

Let

$$G_{ij}(\theta, \hat{\theta}, \hat{y}^{(t)}) = \frac{\partial l_c(f_\theta(x_i), \hat{y}_i^{(t)})}{\partial \theta} \frac{\partial l_{cce}(f_{\hat{\theta}}(x_{m,j}), y_{m,j})}{\partial \hat{\theta}} \tag{5.23}$$

Then Equation 5.5 can be rewritten as

$$\hat{y}^{(t+1)} = \hat{y}^{(t)} + \frac{\alpha\beta}{N_b} \frac{\partial}{\partial \hat{y}} \left(\sum_{i=1}^{N_b} \frac{1}{N_b} \sum_{j=1}^{N_b} G_{ij}(\theta, \hat{\theta}, \hat{y}^{(t)}) \right) \Bigg|_{\hat{y}^{(t)}} \tag{5.24}$$

In this formulation $\frac{1}{N_b} \sum_{i=1}^{N_b} G_{ij}(\theta, \hat{\theta}, \hat{y}^{(t)})$ represents the similarity between the gradient of the j^{th} training sample subjected to θ and the mean gradient computed over the batch of meta-data. Therefore, this will peak when gradients on a training sample and mean gradients over a mini-batch of meta samples are most similar. As a result, taking a gradient step subjected to $\hat{y}$ means finding the optimal label distribution so that produced gradients from training data are similar with gradients from meta-data. This has an effect of emphasizing gradients due to noise free representations and de-emphasizing gradients due to noisy representations. Notice that stochastic gradient descent step is taken on multiplication of previously computed gradients. Hence, this approach of meta-learning is commonly called as *gradients over gradients*.

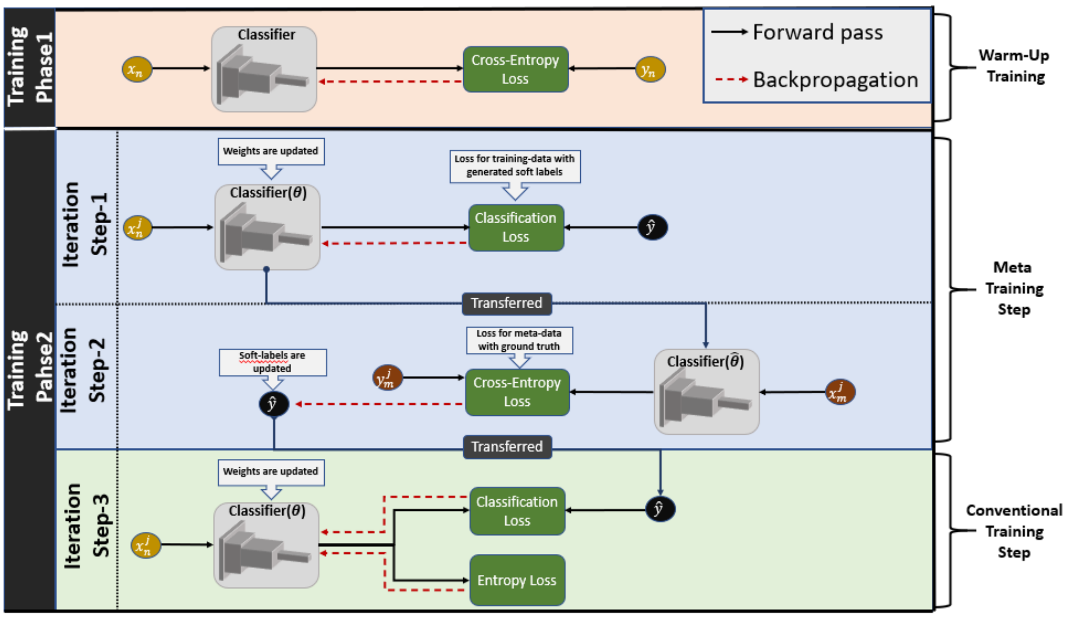


Figure 5.1: The overall framework of the proposed algorithm. x_n, y_n represents the training data and label pair. x_n^j, y_n^j represents batches of training data and label pair. x_m^j, y_m^j represents batches of meta-data and label pair. $\hat{y}$ represents generated soft-labels. The training consists of two phases. In the first phase (warm-up training), the base classifier is trained on noisy data with the conventional cross-entropy loss. This is useful to provide a stable starting point for the second phase. Training phase-2 consists of 3 consecutive steps, which are repeated for each batch of data. In the first iteration step, classification loss for model predictions is calculated with generated soft labels. Then, updated classifier parameters $\hat{\theta}$ are calculated by SGD. In the second iteration step, the cross-entropy loss for meta-data is calculated using the updated classifier parameters $\hat{\theta}$. Then, soft-labels are updated by taking an SGD step on this cross-entropy loss. In the third iteration step, the classifier parameters θ are updated by the classification loss (with newly generated soft labels) and entropy loss.

CHAPTER 6

META-LABEL-NET

The methodology proposed in Chapter 5 successfully handles the noisy labels, but it does not make use of the images' content while producing soft-labels. However, there is an absolute correlation between the data content and its label. Therefore, using the content of the data while creating soft-labels would undoubtedly enhance the performance of the overall system. In Chapter 5, soft-labels are formalized as free differentiable variables that do not depend on anything. Unlike from the previous chapter, the MetaLabelNet algorithm proposed in this chapter interprets soft-labels from the extracted features of the data with the help of a single-layer perceptron network. Therefore, it can produce more reliable labels since additional information coming from the data content is used. Additionally, since a SLP network is used to generate soft-labels instead of free differentiable variables, it provides much stable labels throughout the training. Furthermore, MetaLabelNet does not depend on training data labels. As a result, it can be used for unlabeled data in the same way as the labeled data. This ability extends the use case of the proposed method to much wider applications that consists of both noisily labeled and unlabeled data.

6.1 Training

The full pipeline of the proposed framework is illustrated in Figure 6.1. Similar to the MSLG, the overall training consists of two phases. Phase-1 is the warm-up training, which aims to provide a stable initial point for phase-2 to start on. After the first phase, the main algorithm is employed in the second phase that concludes the training. Each of the training phases and their iteration steps are explained in the following subsections.

6.1.1 Training Phase-1

Justifications given in subsection 5.1.1 are applicable for this algorithm as well. Additionally, different from MSLG, MetaLabelNet needs a feature extractor to map data samples to feature domain in training in phase-2. For that purpose, the trained network at the end of warm-up training is cloned, and its last layer is removed. Then, this clone network, without any further training, is used as the feature extractor in phase-2.

6.1.2 Training Phase-2

This phase is the main training stage of the proposed framework. There are three iteration steps, which are executed on each batch of data consecutively. The first two iterations steps are the meta-training step, in which the parameters of MetaLabelNet are updated. Afterward, in the third iteration step, base classifier parameters are updated by using the soft-labels generated with the updated parameters of the MetaLabelNet. These three steps are executed for each batch of data consecutively. The following subsections explain training steps in detail.

6.1.2.1 Meta Training Step

The meta training step aims to optimize the conventional training step by updating soft-labels. Therefore, it needs feedback from the conventional training step. For that reason, firstly, a dummy conventional training step is conducted on the base classifier with generated soft-labels. Secondly, meta-loss is calculated using the updated classifier and meta-data. Then, meta-loss is backpropagated for MetaLabelNet.

Firstly, data instances are encoded into feature vectors with the help of a feature extractor network $g(.)$.

$$v = g(x) \tag{6.1}$$

Then, depending on these encodings, MetaLabelNet $m_\phi(.)$ generates soft-labels.

$$\hat{y} = m_\phi(v) \tag{6.2}$$

Using these generated labels, posterior model parameters are calculated by taking a stochastic gradient descent step on the classification loss.

$$\hat{\theta} = \theta^{(t)} - \nabla_\theta \mathcal{L}_c(f_\theta(x), \hat{y}^{(t)})\Big|_{\theta^{(t)}} \text{, and} \tag{6.3}$$

$$\mathcal{L}_c(f_\theta(x), \hat{y}^{(t)}) = \frac{1}{N_b}\sum_{i=1}^{N_b} l_c(f_\theta(x_i), \hat{y}_i^{(t)}) \tag{6.4}$$

where $x_i \in \mathcal{D}_n$ and $\hat{y}_i^{(t)} = m_{\phi^{(t)}}(v_i)$ is the corresponding predicted label at time step t. Adapted from [132], the classification loss l_c is chosen as KL-divergence loss as follows

$$l_c = KL(f_\theta(x_i)||\hat{y}_i), \text{where} \tag{6.5}$$

$$KL(f_\theta(x_i)||\hat{y}_i) = \sum_{j=1}^{C} f_\theta^j(x_i) \log\left(\frac{f_\theta^j(x_i)}{\hat{y}_i^j}\right) \tag{6.6}$$

Following, the meta-loss is calculated with the feedback coming from updated parameters.

$$\mathcal{L}_{meta}(f_{\hat{\theta}}(x_m), y_m) = \frac{1}{N_b}\sum_{j=1}^{N_b} l_{cce}(f_{\hat{\theta}}(x_{m,j}), y_{m,j}) \tag{6.7}$$

where $x_{m,j}, y_{m,j} \in \mathcal{D}_m$ and l_{cce} represents the conventional categorical cross-entropy loss. Finally, MetaLabelNet parameters are updated with SGD on the meta loss.

$$\phi^{(t+1)} = \phi^{(t)} - \beta\nabla_\phi \mathcal{L}_{meta}(f_{\hat{\theta}}(x_m), y_m)\Big|_{\phi^{(t)}} \tag{6.8}$$

Mathematical reasoning for the meta-update is further presented in Section 6.3.

6.1.2.2 Conventional Training Step

In this phase, the base network is trained on two losses. The first loss is the classification loss, which ensures that network predictions are consistent with estimated soft-labels.

$$\mathcal{L}_c(f_\theta(x), \hat{y}^{(t+1)}) = \frac{1}{N_b}\sum_{i=1}^{N_b} l_c(f_\theta(x_i), \hat{y}_i^{(t+1)}) \tag{6.9}$$

Notice that this is the same loss formulation with Equation 6.3, but with updated soft-labels $\hat{y}_i^{(t+1)} = m_{\phi^{(t+1)}}(v_i)$. Moreover, inspired from [63], entropy loss is defined as follow.

$$\mathcal{L}_e(f_\theta(x)) = -\frac{1}{N_b}\sum_{i=1}^{N_b}\sum_{j=1}^{C} f_\theta^j(x_i)\log(f_\theta^j(x_i)) \tag{6.10}$$

Entropy loss forces network predictions to peak only at one class. Since the base classifier is trained on predicted soft-labels, which can peak at multiple locations, this is useful to prevent training loop to saturate. Finally, base classifier parameters are updated with SGD on these two losses.

$$\theta^{(t+1)} = \theta^{(t)} - \lambda\nabla_\theta\left(\mathcal{L}_c(f_\theta(x), \hat{y}^{(t+1)}) + \mathcal{L}_e(f_\theta(x))\right)\Big|_{\theta^{(t)}} \tag{6.11}$$

Notice that there are two separate learning rates β and λ for MetaLabelNet and the base classifier. Training continues until the given epoch is reached. Since meta-data is only used for the training of MetaLabelNet, it can be used as validation-data for the base classifier. Therefore, meta-data is used for model selection. The overall training for phase-2 is summarized in Algorithm 3.

6.2 Learning with Unlabeled Data

The presented algorithm in Figure 6.1 uses training data labels y_n only in the training phase-1 (warm-up training). The training phase-2 is totally independent of the training data labels. Therefore, training phase-2 can be used on the unlabeled data in the same way as the labeled data. As a result, if there exists unlabeled data, warm-up training is conducted on the labeled data. Afterwards, the proposed Algorithm 3 is applied to both labeled and unlabeled data in the same manner.

6.3 Reasoning of Meta-Objective

We can rewrite the update term for MetaLabelNet as follow.

$$-\beta\nabla_\phi\mathcal{L}_{meta}(f_{\hat{\theta}}(x_m), y_m)\Big|_{\phi^{(t)}} \tag{6.12}$$

Algorithm 3: Learning with MetaLabelNet

```
Input: Training data $\mathcal{D}_n$, meta-data $\mathcal{D}_m$, batch size $N_b$, phase-1 epoch
       number $e_{phase1}$, phase-2 epoch number $e_{phase2}$,
Output: Base classifier parameters $\theta$, MetaLabelNet parameters $\phi$
$epoch = 0$;
// ----------------- Phase-1 training -----------------
for $epoch < e_{phase1}$ do
    $\theta^{(t+1)} = ConventionalTrain(\theta^{(t)}, \mathcal{D}_n)$    // Warm-up training with noisy data
    $epoch = epoch + 1$
end
// ----------------- Phase-2 training -----------------
for $epoch < e_{phase2}$ do
    foreach batch do
        $\{x\} \leftarrow$ GetBatch($\mathcal{D}_n$,$N_b$)
        $\{x_m, y_m\} \leftarrow$ GetBatch($\mathcal{D}_m$,$N_b$)
        // ----------------- Iteration step-1 -----------------
        $v = g(x)$    // Extract features of data by Equation 6.1
        $\hat{y}^{(t)} = m_\phi(v)$    // Generate soft-labels by Equation 6.2
        $\mathcal{L}_c(f_\theta(x), \hat{y}^{(t)}) = \frac{1}{N_b}\sum_{i=1}^{N_b} l_c(f_\theta(x_i), \hat{y}_i^{(t)})$    // Loss with soft-labels by Equation 6.4
        $\hat{\theta} = \theta^{(t)} - \nabla_\theta \mathcal{L}_c(f_\theta(x), \hat{y}^{(t)})$    // Update $\theta$ by Equation 6.3
        // ----------------- Iteration step-2 -----------------
        $\mathcal{L}_{meta}(f_{\hat{\theta}}(x_m), y_m) =$
            $\frac{1}{N_b}\sum_{j=1}^{N_b} l_{cce}(f_{\hat{\theta}}(x_{m,j}), y_{m,j})$    // Loss with $\hat{\theta}$ on meta-data by Equation 6.7
        $\phi^{(t+1)} = \phi^{(t)} - \beta\nabla_\phi \mathcal{L}_{meta}(f_{\hat{\theta}}(x_m), y_m)$    // Update $\phi$ by Equation 6.8
        // ----------------- Iteration step-3 -----------------
        $v = g(x)$    // Extract features of data by Equation 6.1
        $\hat{y}^{(t)} = m_\phi(v)$    // Generate soft-labels by Equation 6.2
        $\mathcal{L}_c(f_\theta(x), \hat{y}^{(t+1)}) =$
            $\frac{1}{N_b}\sum_{i=1}^{N_b} l_c(f_\theta(x_i), \hat{y}_i^{(t+1)})$    // Loss with new soft-labels by Equation 6.9
        $\mathcal{L}_e(f_\theta(x)) =$
            $-\frac{1}{N_b}\sum_{i=1}^{N_b}\sum_{j=1}^{C} f_\theta^j(x_i)\log(f_\theta^j(x_i))$    // Entropy-loss by Equation 6.10
        $\theta^{(t+1)} = \theta^{(t)} -$
            $\lambda\nabla_\theta\left(\mathcal{L}_c(f_\theta(x), \hat{y}^{(t+1)}) + \mathcal{L}_e(f_\theta(x))\right)$    // Update $\theta$ by Equation 6.11
    end
    $epoch = epoch + 1$;
end
```

$$= -\beta \frac{\partial \mathcal{L}_{meta}(f_{\hat{\theta}}(x_m), y_m)}{\partial \hat{\theta}} \frac{\partial \hat{\theta}}{\partial \phi}\Bigg|_{\phi^{(t)}} \tag{6.13}$$

$$= -\beta \frac{\partial \mathcal{L}_{meta}(f_{\hat{\theta}}(x_m), y_m)}{\partial \hat{\theta}} \frac{\partial}{\partial \phi}\left(-\frac{\partial \mathcal{L}_c(f_\theta(x), \hat{y}^{(t)})}{\partial \theta}\right)\Bigg|_{\phi^{(t)}} \tag{6.14}$$

where $\hat{y}^{(t)} = m_{\phi^{(t)}}(g(x))$. $g(.)$ is a fixed encoder for which no learning occurs. All derivatives of ϕ are taken at time step t. Therefore, the notation $\Big|_{\phi^{(t)}}$ is dropped from now on.

$$= -\frac{\beta}{N_b}\sum_{j=1}^{N_b} \frac{\partial l_{cce}(f_{\hat{\theta}}(x_{m,j}), y_{m,j})}{\partial \hat{\theta}} \frac{\partial}{\partial \phi}\left(-\frac{1}{N_b}\sum_{i=1}^{N_b} \frac{\partial l_c(f_\theta(x_i), \hat{y}_i^{(t)})}{\partial \theta}\right) \tag{6.15}$$

$$= \frac{\beta}{N_b N_b}\frac{\partial}{\partial \phi}\left(\sum_{j=1}^{N_b} \frac{\partial l_{cce}(f_{\hat{\theta}}(x_{m,j}), y_{m,j})}{\partial \hat{\theta}} \sum_{i=1}^{N_b} \frac{\partial l_c(f_\theta(x_i), \hat{y}_i^{(t)})}{\partial \theta}\right) \tag{6.16}$$

$$= \frac{\beta}{N_b N_b}\frac{\partial}{\partial \phi}\left(\sum_{i=1}^{N_b}\sum_{j=1}^{N_b} \frac{\partial l_c(f_\theta(x_i), \hat{y}_i^{(t)})}{\partial \theta} \frac{\partial l_{cce}(f_{\hat{\theta}}(x_{m,j}), y_{m,j})}{\partial \hat{\theta}}\right) \tag{6.17}$$

Let

$$S_{ij}(\theta, \hat{\theta}, \phi^{(t)}) = \frac{\partial l_c(f_\theta(x_i), \hat{y}_i^{(t)})}{\partial \theta} \frac{\partial l_{cce}(f_{\hat{\theta}}(x_{m,j}), y_{m,j})}{\partial \hat{\theta}} \tag{6.18}$$

Then rewrite meta update can be rewritten as

$$\phi^{(t+1)} = \phi^{(t)} + \frac{\beta}{N_b}\frac{\partial}{\partial \phi}\left(\sum_{i=1}^{N_b}\frac{1}{N_b}\sum_{j=1}^{N_b} S_{ij}(\theta, \hat{\theta}, \phi^{(t)})\right) \tag{6.19}$$

In this formulation $\frac{1}{N_b}\sum_{j=1}^{N_b} S_{ij}(\theta, \hat{\theta}, \phi^{(t)})$ represents the similarity between the gradient of the i^{th} training sample subjected to model parameters θ on predicted soft-labels $\hat{y}^{(t)}$ at time step t, and the mean gradient computed over the batch of meta-data. As a result, the similarity is maximized when the gradient of i^{th} training sample is consistent with the mean gradient over a batch of meta-data. Therefore, taking a gradient step subjected to ϕ means finding the optimal parameter set that would give the best $\hat{y} = m_\phi(g(x))$ such that produced gradients from training data are similar to gradients from meta-data.

This approach has three main advantages over MSLG. Firstly, additional information coming from the data features are used. For feature-dependent noises, content of the data is highly correlated with the noisy labels. Therefore, taking data content into consideration helps to enhance the performance. Secondly, $\hat{y}$ is the output of a SLP network instead of free differentiable variables. SLP networks provides a more stable output than free variable. For instance, learning rate of 10^{-3} is enough for MetaLabel-Net while MSLG needs extreme learning rates such as 4000. Thirdly, MetaLabelNet learning framework is independent from training data labels, so it can be used to learn from unlabeled data on top of noisily labeled data.

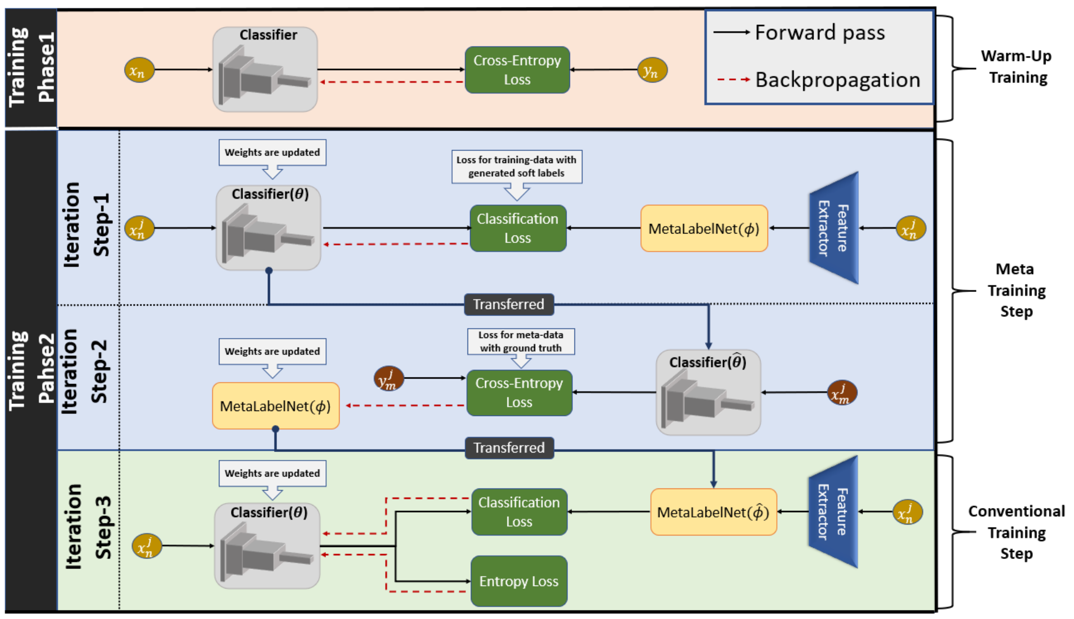


Figure 6.1: The overall framework of the proposed algorithm. x_n, y_n represents the training data and label pair. x_n^j, y_n^j represents batches of training data and label pair. x_m^j, y_m^j represents batches of meta-data and label pair. The training consists of two phases. In the first phase (warm-up training), the base classifier is trained on noisy data with the conventional cross-entropy loss. This is useful to provide a stable starting point for the second phase. Training phase-2 consists of 3 consecutive steps, which are repeated for each batch of data. In the first iteration step, classification loss for model predictions is calculated with soft labels produced by MetaLabelNet. Then, updated classifier parameters $\hat{\theta}$ are calculated by SGD. In the second iteration step, the cross-entropy loss for meta-data is calculated using the updated classifier parameters $\hat{\theta}$. Then, updated MetaLabelNet parameters $\hat{\phi}$ is calculated by taking an SGD step on this cross-entropy loss. In the third iteration step, new soft-labels are produced by updated MetaLabelNet parameters $\hat{\phi}$. Then, classifier parameters θ are updated by the classification loss (with newly generated soft labels) and entropy loss. Feature extractor is the pre-softmax output of the base classifier's duplicate trained at the warm-up phase of the training.

CHAPTER 7

EXPERIMENTS

Proposed MSLG and MetaLabelNet algorithms are tested for five different datasets. Firstly, experiments are conducted on a toy benchmarking dataset CIFAR10 with synthetic label noise. This dataset is used for the quick development and deployment of the proposed algorithms. Experiments are done for varying level of noise ratios in a controlled environment and behaviors of proposed algorithms for varying situations are analyzed. Then, algorithms are tested on three real-world noisy datasets (Clothing1M, Food101N and WebVision). For fair evaluation, the experimental setup is kept same with the baselines. Finally, to show the effectiveness of the algorithms for case study, tests are performed on a real-world noisy medical dataset of ROP plus disease. All results obtained from various datasets with different experimental setups show the superior performance of the proposed algorithms.

Each section in this chapter is organized as follows. Firstly, a detailed description of the dataset is presented. Secondly, experimental setup for the tests is described. Thirdly and finally, the results are presented and discussed.

7.1 CIFAR10

This section describes the experiments conducted on CIFAR10 dataset with synthetic noise.

7.1.1 Dataset Description

CIFAR10 has 60k images for ten different classes. 5k images are separated for the test set and another 2k for meta-set. For fair comparison with other methods, only 50k data for training are used. Training data is corrupted with synthetic label noise.

For synthetic noise, two types of noises are used; uniform noise and feature-dependent noise. For uniform noise, labels are flipped to any other class uniformly with the given error probability. For feature-dependent noise, the algorithm presented in Section 4.2 is employed.

7.1.2 Implementation Details

An eight-layer convolutional neural network with six convolutional layers and two fully connected layers is used as base classifier. The batch size is set as 128. Total training consists of 120 epochs, in which the first 44 epochs are warm-up and the remaining epochs are meta-training. For data augmentation, random vertical and horizontal flips are used. Moreover, images are pad 4 pixels from each side and random crop 32x32 pixels. λ is initialized as 10^{-2} and set to 10^{-3} and 10^{-4} at 40^{th} and 80^{th} epochs. SGD optimizer with 0.9 momentum and 10^{-4} weight decay is used for base classifier. Algorithm specific parameters for MSLG are as given in Table 7.1. Algorithm specific parameters for MetaLabelNet are as follows; $\alpha = 0.5, \beta = 10^{-3}, \gamma = 0.1$. Adam optimizer with learning rate of 10^{-3} is used of meta-network. It is observed that the increasing depth of the meta-network does not contribute to the overall performance. Therefore, a single layer network is used as meta-network with size number of features(256) x number of classes(10).

7.1.3 Results

Algorithm are tested with CIFAR10 under varying level of noise ratios for different types of noises. Since synthetic noise is manually added, we have complete knowledge over the noise. Therefore, the exact noise transition matrix is fed to the forward loss method [48] for baseline comparison, which is not possible in real-world datasets.

Table 7.1: Hyper-parameters for CIFAR10 experiments for MSLG algorithm.

noise ratio	uniform	feature-dependent
20%	$\alpha = 0.5, \beta = 4000$	$\alpha = 0.5, \beta = 4000$
40%	$\alpha = 0.5, \beta = 4000$	$\alpha = 0.5, \beta = 4000$
60%	$\alpha = 0.5, \beta = 2000$	$\alpha = 0.5, \beta = 4000$
80%	$\alpha = 0.5, \beta = 400$	$\alpha = 0.5, \beta = 4000$

Results are presented in Table 7.2 and Table 7.3.

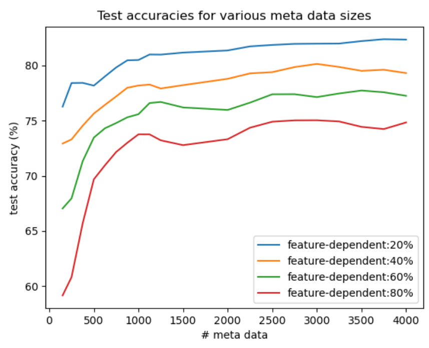
As can be seen, proposed methods beat all baselines with a large margin for all noise rates of the feature-dependent noise. Best results are obtained from MetaLabelNet and second best results are obtained from MSLG. MetaLabelNet manages to get around 71% accuracy even under the extreme case of 80% noise. while MSLG achieves 57%. For uniform noise, proposed algorithms fall shortly behind the best model, but still manages to get comparable results. This can be explained as follow. For uniform noise, noisy samples are totally unrelated to features and true label of the data. Due to the internal robustness of the network, this would result in $f_\theta^j(x_i) \ll \tilde{y}_i^j$ for noisy class j. This would result in large negative gradients Equation 5.16. In the case of feature-dependent noise, noisy labels are related to real label distribution. As a result, network prediction and noisy label is more similar $f_\theta^j(x_i) < \tilde{y}_i^j$ for noisy class j. This would result in smaller gradients Equation 5.16. Therefore, when noise is random, gradients due to noisy class may overcome the gradients due to true class, which presents a more challenging framework for stabilization of robust learning. Therefore, proposed frameworks are slightly less robust to random noise. This is an advantage for real-world scenarios since noisy labels are commonly related to data attributes. This is further showed on real-world noisy dataset in the next section.

Both algorithm are tested with changing number of meta-data samples. As expected, accuracy increases with increasing number of meta-data. However, as can be seen from Figure 7.1, only small amount of meta-data is required for the proposed frameworks. For CIFAR10 dataset with 50k noisy training samples, 1k meta-data (2% of training data) achieves approximately the top result. Moreover, as presented in Figure 7.1, number of required meta-data is independent of the noise ratio.

Table 7.2: Test accuracies for CIFAR10 dataset with varying level of uniform noise. Results are averaged over 4 runs.

noise type	uniform			
noise ratio (%)	20	40	60	80
Cross Entropy	82.55±0.80	76.31±0.75	65.94±0.44	38.19±0.81
Symmetric-CE[119]	81.36±2.27	78.94±2.22	72.20±2.24	51.47±1.58
Generalized-CE[118]	84.98±0.30	**81.65±0.30**	**74.59±0.46**	42.53±0.24
Bootstrap [89]	82.76±0.36	76.66±0.72	66.33±0.30	38.35±1.83
Forward Loss[48]	83.24±0.37	79.69±0.49	71.41±0.80	31.53±2.75
Joint Opt.[63]	83.64±0.50	78.69±0.62	68.83±0.22	39.59±0.77
PENCIL[64]	83.86±0.47	79.01±0.62	71.53±0.39	46.07±0.75
Co-Teaching[91]	**85.82±0.22**	80.11±0.41	70.02±0.50	39.83±2.62
MLNT[128]	83.32±0.45	77.59±0.85	67.44±0.45	38.83±1.76
Meta-Weight[97]	83.59±0.54	80.22±0.16	71.22±0.81	45.81±1.78
MSLG	83.03±0.44	78.28±1.03	71.30±1.54	**52.43±1.26**
MetaLabelNet	83.35±0.17	79.03±0.26	70.60±0.86	50.02±0.44

(a) MSLG



(b) MetaLabelNet

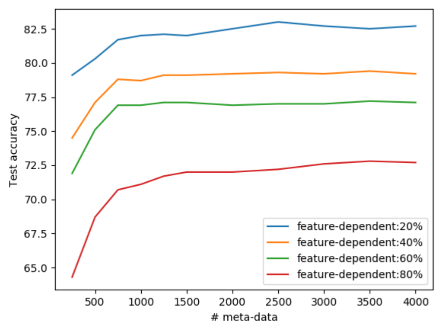


Figure 7.1: Test accuracies for different numbers of meta-data.

Table 7.3: Test accuracies for CIFAR10 dataset with varying level of feature-dependent noise. Results are averaged over 4 runs.

noise type	feature-dependent			
noise ratio (%)	20	40	60	80
Cross Entropy	81.75±0.39	71.86±0.69	69.78±0.71	23.18±0.55
Symmetric-CE[119]	74.45±2.97	63.71±0.58	fail	fail
Generalized-CE[118]	81.43±0.45	72.35±0.51	66.60±0.43	fail
Bootstrap [89]	81.59±0.61	72.18±0.86	69.50±0.29	23.05±0.64
Forward Loss[48]	77.17±0.86	69.46±0.68	36.98±0.73	fail
Joint Opt.[63]	81.83±0.51	74.06±0.57	71.74±0.63	44.81±0.93
PENCIL[64]	81.82±0.41	75.18±0.50	69.10±0.24	fail
Co-Teaching[91]	81.07±0.25	72.73±0.61	68.08±0.42	18.77±0.07
MLNT[128]	82.07±0.76	73.90±0.34	69.16±1.13	22.85±0.45
Meta-Weight[97]	81.36±0.54	72.52±0.51	67.59±0.43	22.10±0.69
MSLG	82.22±0.57	77.62±0.98	73.08±1.58	57.30±7.40
MetaLabelNet	**83.00±0.41**	**80.42±0.51**	**78.42±0.36**	**76.57±0.33**

Figure 7.2 shows the difference in the learning regime of MSLG and MetaLabelNet. As can be seen, MetaLabelNet provides much more stabilized labels throughout the epochs. This is due to two factors. Firstly, MetaLabelNet interprets soft-labels from data features while MSLG freely updates labels as a differentiable variable of the framework. Secondly, MetaLabelNet uses a small MLP network to predict labels while MSLG directly updates with unbounded gradients coming from meta-objective. Therefore, MSLG needs to use extreme learning rates around 4000 while MetaLabelNet uses learning rate around 10^{-3}. As a result, MetaLabelNet provides a much more stabilized update on label predictions. Additionally, MSLG can only generate soft-labels for labeled data since there is no label generation algorithm trained. On the other hand, once trained, MetaLabelNet can produce soft-labels for unlabeled data. Therefore, MetaLabelNet can work with both noisily labeled and unlabeled data at the same time.

The efficiency of the MetaLabelNet is demonstrated in the existence of unlabeled data in Figure 7.3. Labels of indicated number of samples are removed from the training set. Warm-up training is conducted on labeled part of the data. As illustrated, MetaLabelNet can give top accuracy even up to point where half of the data is unlabeled.

7.2 Clothing1M

This section describes the experiments conducted on real-world noisy dataset Clothing1M.

7.2.1 Dataset Description

Clothing1M is a large-scale dataset with one million images collected from the web [18]. It has images of clothings from 14 classes. Labels are constructed from surrounding texts of images and are estimated to have a noise rate of around 40%. There exists 50k, 14k and 10k additional verified images for training, validation and test set. 14k is used for validation set as meta-data and 10k test samples to evaluate the classifier's final performance. 50k clean training samples are not used in any part of training.

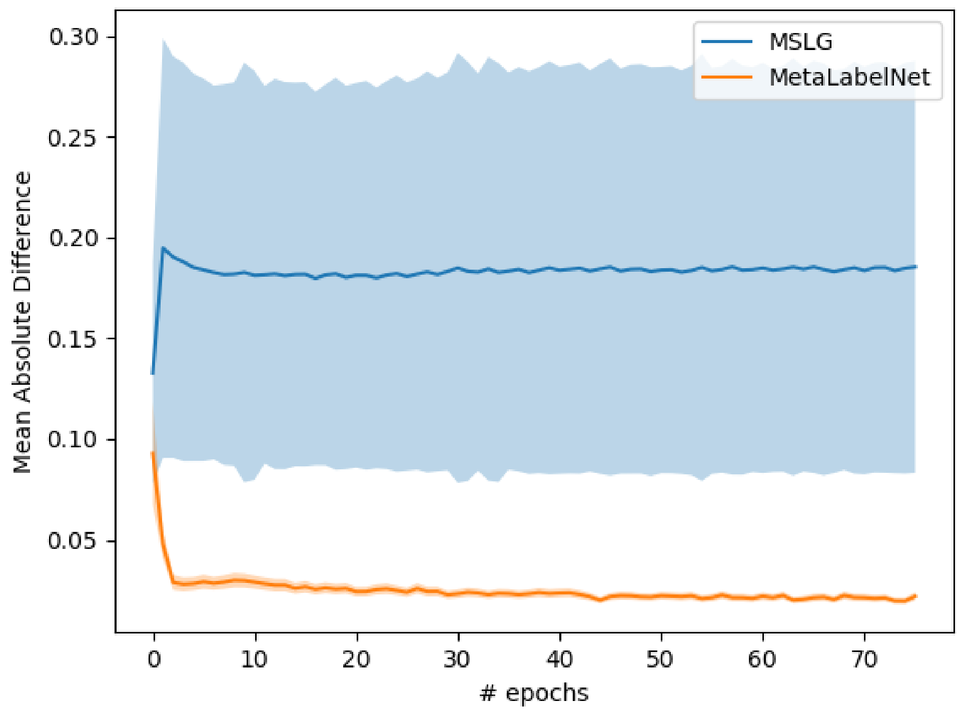


Figure 7.2: Colored lines represent the mean absolute difference among generated soft labels for consecutive epochs. Shaded regions are the variance of the differences. As can be seen MSLG generates highly unstabilized label predictions compared to MetaLabelNet.

7.2.2 Implementation Details

In order to have a fair comparison with the works from the literature, the widely used setup of ResNet-50 [3] architecture pre-trained on ImageNet is implemented. Batch size is set to 32. λ is set to 10^{-3} for the first 5 epochs and to 10^{-4} for the second 5 epochs. Total training consists of 10 epochs, in which the first epoch is warm-up and the rest is meta-training. All images are resized to 256x256, and then central 224x224 pixels are taken. SGD optimizer with 0.9 momentum and 10^{-4} weight decay is used for base classifier. For MSLG algorithm hyper-parameters are set as $\alpha = 0.1, \beta = 100$ and for MetaLabelNet algorithm hyper-parameters are set as $\alpha = 0.5, \beta = 10^{-3}, \gamma = 0.1$. Adam optimizer with learning rate of 10^{-3} is used of meta-network. It is observed that the increasing depth of the meta-network does

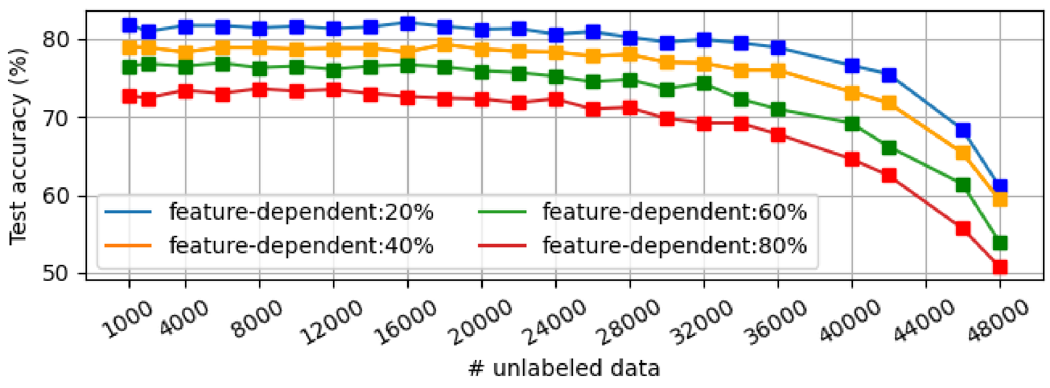


Figure 7.3: Test accuracies of MetaLabelNet for different level of feature-dependent noises for varying sizes of unlabeled data.

not contribute to the overall performance. Therefore, a single layer network is used as meta-network with size number of features(2048) x number of classes(10).

7.2.3 Results

Clothing1M is a widely used benchmarking dataset to evaluate the performance of proposed algorithms in the presence of noisy labels. State of the art results from the literature are presented in Table 7.4, where proposed algorithms managed to outperformed all baselines. MSLG achieves 76.02% test accuracy, which is 2.3% higher than the closest state of the art.On top of that, MetaLabelNet achieves 77.9% test accuracy, which is 1.9% higher than MSLG.

Table 7.5 presents the effectiveness of the MetaLabelNet in the existence of unlabeled data. Clothing1M is a highly unbalanced dataset, so removing labels of data randomly would cause undesirable performance degradation due to data imbalance. Therefore, a new balanced dataset consists of 260k images is constructed by picking samples from 1M training data. Afterward, indicated number of labels are removed from the dataset and training is constructed on this new dataset with labeled and unlabeled instances. For comparison an additional model is trained with classical cross entropy method only on labeled data samples. As presented, MetaLabelNet achieves superior performance even there are unlabeled data instances in the dataset. Even in the extreme case of 225k unlabeled data (which is 87% of the training data) MetaL-

Table 7.4: Test accuracy percentages on Clothing1M dataset. All results are taken from the corresponding paper.

method	**accuracy**
Joint Optimization [63]	72.23
MetaCleaner [152]	72.50
SafeGuarded [56]	73.07
MLNT [128]	73.47
PENCIL [64]	73.49
Meta-Weight Net [97]	73.72
NoiseRank [78]	73.77
Anchor points[55]	74.18
CleanNet [102]	74.69
DivideMix [1]	74.76
MSLG	76.02
MetaLabelNet	77.90

abelNet manages to achieve 74.1% accuracy. In the case of 175k unlabeled and 85k labeled data, MetaLabelNet outperforms all state-of-the-art baselines.

Table 7.5: Test accuracies for varying number of unlabeled data. Subset of Clothing1M dataset is used, which is balanced for each class.

# unlabeled data	50k	75k	100k	125k	150k	175k	200k	225k
Cross Entropy	71.40	70.52	70.35	70.66	69.42	69.79	68.47	68.73
MetaLabelNet	76.4	76.8	76.4	76.0	75.7	75.5	74.4	74.1

7.3 Food101N

7.3.0.1 Dataset Description

Food101N is an image dataset containing about 310k images of food recipes classified in 101 classes [102]. It shares the same classes with Food101 dataset but has much more noisy labels, which is estimated to be around 20%. It has 53k verified training and 5k verified test images. 15k samples from verified training samples are used as meta-dataset.

7.3.1 Implementation Details

We used the same setup and parameter set from Clothing1M. Only difference are at the following parameters of MSLG; $\lambda = 0.5, \beta = 1500$

7.3.2 Results

In order to further test algorithms under real-world noisy label data, tests are conducted on Food101N dataset too. Since none of the baselines provided results on Food101N dataset, all results are taken from our own implementations. There are excessively large number of classes (101) in the dataset, hence some methods fail to succeed. For example, methods depending on noise transition matrix fail since the matrix becomes intractably large. Only the results of methods with fair performances are presented. This dataset has a much smaller noise ratio (20%), as a result all algorithms results around similar accuracies with straight forward training with cross-entropy loss. Therefore, there are no big gaps among top accuracies, but still as presented in Table 7.6, presented algorithms manage to get best accuracy in this dataset too.

Table 7.6: Test accuracy percentages for Food101N dataset. All values in the table are obtained from our own implementations.

method	**accuracy**
Generalized CE [118]	71.60
Joint Optimization [63]	76.12
Meta-Weight Net [97]	76.14
Bootstrap [89]	78.03
PENCIL [64]	78.26
Co-Teaching [91]	78.95
MSLG	79.06
MetaLabelNet	80.21

7.4 WebVision

This section describes the experiments conducted on real-world noisy dataset WebVision.

7.4.1 Dataset Description

WebVision 1.0 dataset [19] consists of 2.4 million images crawled from Flickr website and Google Images search. As a result, it has many real-world noisy labels. It has images from 1000 classes that are same with ImageNet ILSVRC 2012 dataset [173]. In order to have fair comparison with previous works, the same setup with [94] is used as follows. Only Google subset of data is used and among these data samples only from the first 50 classes are picked. This subset contains 2.5k verified test samples, from which 1k is used as meta-data and 1.5k as test set.

7.4.2 Implementation Details

Following the previous works [94, 1], inception-resnet v2 [174] network architecture with random initialization is used. Batch size is set to 16. λ is set to 10^{-3} for the first

Table 7.7: Test accuracies on WebVision dataset. Baseline results are taken from the [1]

method	Top1	Top5
Forward Loss [48]	61.12	82.68
Decoupling [90]	62.54	84.74
D2L [140]	62.68	84.00
MentorNet [85]	63.00	81.40
Co-Teaching [91]	64.58	85.20
Iterative-CV [94]	65.24	85.34
MSLG	65.45	85.72
MetaLabelNet	67.10	87.48

50 epochs and to 10^{-4} for the second 50 epochs. Total training consists of 100 epochs, in which the first 44 epochs are warm-up and the rest is meta-training. All images are resized to 320x320, and then central 299x299 pixels are taken. Furthermore, random horizontal flip is applied. SGD optimizer with 0.9 momentum and 10^{-4} weight decay is used for base classifier. For MSLG algorithm hyper-parameters are set as $\alpha = 0.5, \beta = 10^{-3}, \gamma = 0.1$ and for MetaLabelNet algorithm hyper-parameters are set as $\alpha = 0.5, \beta = 10^{-3}, \gamma = 0.1$. Adam optimizer with learning rate of 10^{-3} is used of meta-network. It is observed that the increasing depth of the meta-network does not contribute to the overall performance. Therefore, a single layer network is used as meta-network with size number of features(1536) x number of classes(50).

7.4.3 Results

Due to its noisiness, WebVision is another popular dataset for testing robustness to label noise. State of the art results from the literature are presented in Table 7.7, where proposed algorithms managed to outperformed all baselines.

7.5 Retinopaty of Prematurity

In order to show the efficiency of the proposed algorithms for a real-world case, ROP dataset with extreme label noise ratio is gathered. This chapter describes the experiments conducted on this real-world medical imaging dataset. This work is previously published in [175], and the content is highly correlated with the original paper.

7.5.1 Dataset Description

Dataset consists of 1947 retina images with 640x480 resolution collected from control subjects. Each patient belongs to either one of two ROP stages (plus and pre-plus) or normal category which indicates no disease. Each image is labeled by the same three experts to one of the following categories: normal, pre-plus and plus. From these 1947 images, only on 622 images all three experts gave the same label, which is 32% of the dataset. 200 images are selected as meta-data and 300 images as test-data from these 622 images. Remaining 1447 images are used for training. Majority voting is used on training data labels to determine the label. For preprocessing, all images are resized to 256x256 and the center crop 224x224. Random horizontal flip is used for data augmentation purposes.

7.5.2 Implementation Details

ResNet50 architecture with model parameters pre-trained on the ImageNet dataset is used as base classifier. Gathered ROP dataset is considerably small; therefore, it is beneficial to apply transfer learning by further pre-training network on a different but similar dataset. For that purpose diabetic retinopathy dataset that has 35k high-resolution retina images [176] is used. A clinician has rated the existence of diabetic retinopathy on a scale of 0 to 4 as follows: no diabetic retinopathy (0), mild (1), moderate (2), severe (3), proliferative DR (4). Since main purpose of this work is not to classify DR, but rather pre-train network to learn useful representation mapping, dataset is converted to a binary classification task as: diabetic retinopathy (0) and no diabetic retinopathy (1). This dataset consists of high-resolution images with varying sizes. Therefore, all images are first resized to 1024x1024 and then cropped 224x224 around the center of the retina image. On the resulting dataset, model is trained for one epoch. Afterward, the final layer of the classifier is replaced with randomly initialized 2048x4 fully connected layer and softmax layer to match collected ROP dataset labels.

After pre-training on DR dataset, the proposed algorithm is employed. Stochastic gradient descent optimizer with momentum 0.9 and weight decay 10^{-4} is used for base classifier. Learning rate is initialized as 10^{-3} and set it to 10^{-4} and 10^{-5} at 10^{th} and 20^{th} epochs. Total training consists of 30 epochs, in which first 10 epochs are warm-up training and rest is meta-training. During the whole training batch size of 16 is used. For MSLG the following hyper-parameters set is used; $K = 10, \alpha = 0.5, \beta = 4000$. For MetaLabelNet, it is observed that the increasing depth of the meta-network does not contribute to the overall performance. Therefore, a single layer network is used as meta-network with size number of features(2048) x number of classes(10). Adam optimizer with 10^{-4} learning rate and 10^{-5} weight decay is used for meta network. Hyper-parameters are set as $\alpha = 0.5, \beta = 10^{-3}, \gamma = 0.1$.

7.5.3 Results

In order to show the effectiveness of MetaLabelNet, its performance is compared to classical learning with cross-entropy loss. Results are provided in Table 7.8. First, the model is directly trained on the ROP dataset without any pre-training on the DR dataset. This run achieved 86.3% test accuracy, which is moderate but the worst performance in the leaderboard. Secondly, the network is pre-trained on the DR dataset and then continued to conventional training with cross-entropy on the ROP dataset. This run resulted in 90.3%, which is 4% more than the previous run. Increase in the performance shows the effectiveness of pre-training on the DR dataset. Thirdly, MSLG algorithm is employed, where it manages 91.4% accuracy. Finally, MetaLabelNet is employed that gives the best performance with 93.2%. Another important observation is, in conventional cross-entropy loss both networks manage to get 100% training accuracy. Considering that labels are extremely noisy, this is an undesired behavior that means the network is overfitting the data. On the other hand, for Met-

Table 7.8: Train and test accuracies on ROP dataset.

Method	**Train Accuracy**	**Test Accuracy**
Cross Entropy (no pre-train)	100.0%	86.3%
Cross Entropy (pre-train on DR)	100.0%	90.3%
MSLG	86.23%	91.4%
MetaLabelNet	**70.1%**	**94.3%**

aLabelNet, training accuracy is stuck at 70.1%. Therefore, it can be concluded that MetaLabelNet prevents model to overfit the noise. Moreover, when the training accuracy with generated labels by meta-network is 81.2%. This also indicates that generated labels have a cleaner representation of the data since it is higher than training accuracy.

CHAPTER 8

CONCLUSION

In this thesis, the phenomenon of learning from noisily labeled data is investigated thoroughly. It is shown that label noise is an important obstacle to deal with in order to achieve desirable performance from real-world datasets. In spite of its importance for supervised learning in practical applications, it is also an important step to collect datasets from the web [177, 178], design networks that can learn from unlimited web data with no human supervision [35, 36, 37, 38]. Furthermore, beside image classification, there are more fields where dealing with mislabeled instances is important, such as generative networks [179, 180], semantic segmentation [27, 28, 29], sound classification [181] and more. All these factors make dealing with label noise an important step through self-sustained learning systems.

Firstly, types of noisy labels and the cause of their existence in the datasets is explained in Chapter 2. Furthermore, preliminary knowledge on the meta-learning techniques is presented, which is essential to comprehend algorithms presented in the following sections.

Secondly, an elaborative investigation of the works from the literature is presented in Chapter 3. In the presence of label noise, main challenge is to learn optimal estimator for $p(y|x)$ while there is only access is to noisy set $(x, \tilde{y})$. There are two major lines of approaches to achieve this. The first line of work aims to model underlying noise structure and learns probability distribution conditioned on this knowledge. On the other hand, the second line of work aims to directly learn $p(y|x)$ with inherently noise-tolerant algorithms. Noise model based approaches decouple classification and noise estimation algorithms. There are two questions in this case: how to find out noise structure and how to use it. The first question is the common challenge for all approaches in this category, while the second question determines its type of approach within this class of algorithms. Some works take true labels y as a hidden variable, where given noisy labels $\tilde{y}$ is a corrupted version of y by a noisy channel. In order to force the network to estimate y, network predictions are mapped to noisy labels by a noisy channel model, and *corrected loss* is backpropagated [48]. A widely used approach is to assume class-dependent noise and construct a *noise transition matrix* either explicitly [48] or iteratively [53]. Alternatively, more complex noisy channel characteristics can be modeled by neural networks [18]. Instead of estimating $p(\tilde{y}|y)$ (or $p(\tilde{y}|y, x)$ for feature-dependent noise) as in noisy channel case, alternative approach is to estimate $(y|\tilde{y}, x)$. That is to find noise-free labels from the given set of noisy data. This is called as label noise cleansing and main trends separated as using a subset of the dataset with clean labels [61] or using with just noisy labels [63, 64]. Alternatively, some works suggest removing suspicious samples [77] or their noisy labels [79], instead of correcting them. Even though it results in a loss of information, pruning from noise can still increase the performance. As an extended version of dataset pruning, one can dynamically change the dataset for each iteration. This type of algorithms are called as sample choosing. Some works under this category rank instances according to a determined metric so that data is feed in order of rank [84, 85]. Others use multiple networks to choose samples for their pair [91]. Instead of just choosing samples, one can aim to find a weighting scheme for data instances to minimize the expected risk in the target domain, which is the clean distribution $\mathcal{D}$. The weighting scheme can either be calculated statistically [98] or meta-learning paradigm can be used to learn the best weighting scheme [95, 97]. In some cases, multiple annotators label the data, which results in a multi-labeled dataset. This can be seen as a multiple noisy channel problem in which each noisy channel represents separate labelers. Then noise characteristics for each labeler can be modeled with noise transition matrix [24, 108] or separate networks [23]. The second major line of approaches aims to achieve noise tolerance without explicitly modeling the noise. Most of the works in this category focus on robust losses, such that loss function would result in similar loss values for both clean and noisy data. From this perspective, widely used categorical cross entropy is known to be vulnerable to noise. Therefore, various modified versions of categorical cross entropy are proposed in the literature [118, 119]. Even though these losses help increasing robustness, they are still affected by noise. Another promising direction is the use of the meta-learning paradigm. In addition to classical risk minimization, these algorithms define a meta-objective that would result in a more tolerant model in the end. These tasks can be in a broad range such as finding noise robust weight initialization [128], knowledge distillation from clean data [129], and finding optimal sample weighting scheme [95]. Since noise is seen as an anomaly, some works also focus on regularizers to prevent overfit [140]. Finally, some ensemble learning techniques are proposed to compensate for the extreme vulnerability of boosting to noise, but they are mostly shadowed by other approaches.

Thirdly, a novel synthetic feature-dependent label noise generation algorithm is proposed in Chapter 4. This is an essential step for developing noise robust algorithms. Because noisy real-world datasets are generally huge, development process is considerably slow. Furthermore, in real-world noisy datasets, noisy labels are ambiguous. So it is hard to investigate the noise affected characteristic of the deep network since ground truth labels are missing for a big part of the dataset. As a result, adding artificial noise to the toy dataset is handy for quick deployment and testing of the algorithms. In order to make artificial noise as realistic as possible, an algorithm that uses the knowledge distillation technique [45] to generate feature-dependent noisy labels is presented.

Fourthly, label noise robust learning algorithm, which is powered by meta-learning techniques, is presented in Chapter 5. The proposed algorithm MSLG is based on the following simple assumption: *optimal model parameters learned with noisy training data should minimize the cross-entropy loss on clean meta-data.* The proposed method iteratively generates soft-labels for each data and uses these labels to train the base classifier. Soft-labels are produced according to a meta-objective that is to minimize the loss on a small amount of clean meta-data. MSLG can also be seen as a label cleaning approach since it iteratively applies label correction. Nevertheless, it differs from conventional label noise cleansing approaches in a way that it is not searching for clean hard-labels but rather searching for optimal labels in soft-label space to minimize meta-objective. Therefore, propagated soft labels are not necessarily clean labels, but optimal labels for meta-objectives in a different label space.

Fifthly, the algorithm proposed in Chapter 5 is further improved by considering data features while generating soft-labels. In MSLG, label predictions are formularized as free differentiable variables and they are updated by the gradients coming from the meta-objective. However, there is a strong correlation between the data content and its corresponding optimal soft-label. Therefore, using data features while generating soft-labels would enhance the performance of the system. Such an algorithm is presented in Chapter 6. Each instance is mapped to a feature vector by a feature extractor network, and then a small single-layer perceptron network is trained to generate soft labels depending on these features. This improvement enables a much stable learning loop for the base classifier. Furthermore, proposed algorithm MetaLabelNet can be applied to unlabeled data in combination with noisily labeled data. To the best of the knowledge of the author, there is no algorithm proposed to deal with such a problem setup in the literature.

Sixthly and finally, proposed algorithms are tested for various datasets. Algorithms are first verified on CIFAR10 dataset with synthetic label noise. Afterward, extensive experiments are conducted on real-world noisy datasets, which are widely used as benchmarks in the literature. Results show that the proposed algorithm beats the state of the art baselines with a large margin in all datasets. Moreover, the developed algorithm is applied to a case study of real-world noisy medical dataset of ROP disease. It is observed that the presented algorithm successfully manages to classify stages of ROP disease from a small and noisy dataset. All these results show the superiority of the proposed approaches.

There are several possible future research directions on top of algorithms presented in this thesis. Both MSLG and MetaLabelNet requires a clean subset of data to be used as meta data. This requirement can be eliminated by designing an additional algorithm to pick samples, within the noisy training data, to be used as meta data. There are two constraints while picking samples; they should be noise free and they should be informative enough to produce useful meta-gradients. However, these two constraints generally conflicts with each other. Hard informative samples are tend to be seen as outlier and behaved as noisy data. Therefore, a fruitful future research direction is to design a sample picking algorithm that watches aforementioned constraints. Then, meta dataset can be constructed with this algorithm and requirement for predefined meta dataset can be eliminated. Another possible future research direction is to extend presented algorithms beyond the noisy label problem domain and be merged with self-supervised learning techniques. In such a setup, one can put effort into picking up the most informative samples. Later on, by using these samples as meta-data, the proposed meta-learning framework can be used as a performance booster for the self-supervised learning algorithm at hand.

REFERENCES

[1] J. Li, R. Socher, and S. C. Hoi, "Dividemix: Learning with noisy labels as semi-supervised learning," in *International Conference on Learning Representations*, 2020.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[4] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580–587, 2014.

[5] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, pp. 91–99, 2015.

[6] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*, pp. 21–37, Springer, 2016.

[7] G. Lin, C. Shen, A. Van Den Hengel, and I. Reid, "Efficient piecewise training of deep structured models for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3194–3203, 2016.

[8] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for se-

mantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440, 2015.

[9] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," in *International Conference on Learning Representations*, 2017.

[10] D. Arpit, S. Jastrzebski, N. Ballas, D. Krueger, E. Bengio, M. S. Kanwal, T. Maharaj, A. Fischer, A. Courville, Y. Bengio, *et al.*, "A closer look at memorization in deep networks," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 233–242, JMLR. org, 2017.

[11] X. Zhu and X. Wu, "Class noise vs. attribute noise: A quantitative study of their impacts," *Artif. Intell. Rev.*, vol. 22, p. 177–210, Nov. 2004.

[12] B. Frénay and M. Verleysen, "Classification in the presence of label noise: a survey," *IEEE transactions on neural networks and learning systems*, vol. 25, no. 5, pp. 845–869, 2014.

[13] D. F. Nettleton, A. Orriols-Puig, and A. Fornells, "A study of the effect of different types of noise on the precision of supervised learning techniques," *Artificial intelligence review*, vol. 33, no. 4, pp. 275–306, 2010.

[14] M. Pechenizkiy, A. Tsymbal, S. Puuronen, and O. Pechenizkiy, "Class noise and supervised learning in medical domains: The effect of feature extraction," in *19th IEEE Symposium on Computer-Based Medical Systems (CBMS'06)*, pp. 708–713, IEEE, 2006.

[15] B. Li, Y. Wang, A. Singh, and Y. Vorobeychik, "Data poisoning attacks on factorization-based collaborative filtering," in *Advances in Neural Information Processing Systems*, pp. 1893–1901, 2016.

[16] J. Steinhardt, P. W. Koh, and P. Liang, "Certified defenses for data poisoning attacks," in *Advances in Neural Information Processing Systems*, vol. 2017-Decem, pp. 3518–3530, 2017.

[17] D. Angluin and P. Laird, "Learning from noisy examples," *Machine Learning*, vol. 2, no. 4, pp. 343–370, 1988.

[18] T. Xiao, T. Xia, Y. Yang, C. Huang, and X. Wang, "Learning from massive noisy labeled data for image classification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2691–2699, 2015.

[19] W. Li, L. Wang, W. Li, E. Agustsson, and L. Van Gool, "Webvision database: Visual learning and understanding from web data," *arXiv preprint arXiv:1708.02862*, 2017.

[20] A. Torralba, R. Fergus, and W. T. Freeman, "80 million tiny images: A large data set for nonparametric object and scene recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 30, no. 11, pp. 1958–1970, 2008.

[21] L. P. Garcia, J. Lehmann, A. C. de Carvalho, and A. C. Lorena, "New label noise injection methods for the evaluation of noise filters," *Knowledge-Based Systems*, vol. 163, pp. 693–704, 2019.

[22] G. Algan and İ. Ulusoy, "Label noise types and their effects on deep learning," *arXiv preprint arXiv:2003.10471*, 2020.

[23] M. Y. Guan, V. Gulshan, A. M. Dai, and G. E. Hinton, "Who said what: Modeling individual labelers improves classification," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[24] A. Khetan, Z. C. Lipton, and A. Anandkumar, "Learning from noisy singly-labeled data," *arXiv preprint arXiv:1712.04577*, 2017.

[25] Y. Dgani, H. Greenspan, and J. Goldberger, "Training a neural network based on unreliable human annotation of medical images," in *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, pp. 39–42, IEEE, 2018.

[26] C. Xue, Q. Dou, X. Shi, H. Chen, and P.-A. Heng, "Robust learning at noisy labeled medical images: Applied to skin lesion classification," in *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)*, pp. 1280–1283, IEEE, 2019.

[27] Z. Lu, Z. Fu, T. Xiang, P. Han, L. Wang, and X. Gao, “Learning from weak and noisy labels for semantic segmentation,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 3, pp. 486–500, 2016.

[28] Y. Zhu, K. Sapra, F. A. Reda, K. J. Shih, S. Newsam, A. Tao, and B. Catanzaro, “Improving semantic segmentation via video propagation and label relaxation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8856–8865, 2019.

[29] D. Acuna, A. Kar, and S. Fidler, “Devil is in the edges: Learning semantic boundaries from noisy annotations,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 11075–11083, 2019.

[30] P. Welinder, S. Branson, P. Perona, and S. J. Belongie, “The multidimensional wisdom of crowds,” in *Advances in neural information processing systems*, pp. 2424–2432, 2010.

[31] Y. Cha and J. Cho, “Social-network analysis using topic models,” in *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pp. 565–574, ACM, 2012.

[32] Y. Wang, Y. Rao, X. Zhan, H. Chen, M. Luo, and J. Yin, “Sentiment and emotion classification over noisy labels,” *Knowledge-Based Systems*, vol. 111, pp. 207–216, 2016.

[33] Y. Aït-Sahalia, J. Fan, and D. Xiu, “High-frequency covariance estimates with noisy and asynchronous financial data,” *Journal of the American Statistical Association*, vol. 105, no. 492, pp. 1504–1517, 2010.

[34] F. Schroff, A. Criminisi, and A. Zisserman, “Harvesting image databases from the web,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 4, pp. 754–766, 2010.

[35] R. Fergus, L. Fei-Fei, P. Perona, and A. Zisserman, “Learning object categories from internet image searches,” *Proceedings of the IEEE*, vol. 98, no. 8, pp. 1453–1466, 2010.

[36] X. Chen, A. Shrivastava, and A. Gupta, "NEIL: Extracting visual knowledge from web data," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1409–1416, 2013.

[37] S. K. Divvala, A. Farhadi, and C. Guestrin, "Learning everything about anything: Webly-supervised visual concept learning," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 3270–3277, 2014.

[38] A. Joulin, L. van der Maaten, A. Jabri, and N. Vasilache, "Learning visual features from large weakly supervised data," in *European Conference on Computer Vision*, pp. 67–84, Springer, 2016.

[39] J. Krause, B. Sapp, A. Howard, H. Zhou, A. Toshev, T. Duerig, J. Philbin, and L. Fei-Fei, "The unreasonable effectiveness of noisy data for fine-grained recognition," in *European Conference on Computer Vision*, pp. 301–320, Springer, 2016.

[40] J. De Fauw, J. R. Ledsam, B. Romera-Paredes, S. Nikolov, N. Tomasev, S. Blackwell, H. Askham, X. Glorot, B. O'Donoghue, D. Visentin, *et al.*, "Clinically applicable deep learning for diagnosis and referral in retinal disease," *Nature medicine*, vol. 24, no. 9, p. 1342, 2018.

[41] V. Gulshan, L. Peng, M. Coram, M. C. Stumpe, D. Wu, A. Narayanaswamy, S. Venugopalan, K. Widner, T. Madams, J. Cuadros, *et al.*, "Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs," *Jama*, vol. 316, no. 22, pp. 2402–2410, 2016.

[42] L. Xie, J. Wang, Z. Wei, M. Wang, and Q. Tian, "Disturblabel: Regularizing cnn on the loss layer," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4753–4762, 2016.

[43] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. De Freitas, "Learning to learn by gradient descent by gradient descent," in *Advances in Neural Information Processing Systems*, pp. 3981–3989, 2016.

[44] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1126–1135, JMLR. org, 2017.

[45] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[46] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, and K. Murphy, "Progressive neural architecture search," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 19–34, 2018.

[47] G. Algan and I. Ulusoy, "Image classification with deep learning in the presence of noisy labels: A survey," *Knowledge-Based Systems*, vol. 215, p. 106771, 2021.

[48] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1944–1952, 2017.

[49] 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*, pp. 10456–10465, 2018.

[50] X. Chen and A. Gupta, "Webly supervised learning of convolutional networks," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1431–1439, 2015.

[51] A. J. Bekker and J. Goldberger, "Training deep neural-networks based on unreliable labels," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2682–2686, IEEE, 2016.

[52] J. Goldberger and E. Ben-Reuven, "Training deep neural-networks using a noise adaptation layer," in *Proceedings of International Conferance on Learning Representations (ICLR)*, 2017.

[53] S. Sukhbaatar and R. Fergus, "Learning from noisy labels with deep neural networks," *arXiv preprint arXiv:1406.2080*, 2014.

[54] S. Sukhbaatar, J. Bruna, M. Paluri, L. Bourdev, and R. Fergus, "Training convolutional networks with noisy labels," in *International Conference on Learning Representations*, 2015.

[55] 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*, pp. 6835–6846, 2019.

[56] J. Yao, H. Wu, Y. Zhang, I. W. Tsang, and J. Sun, "Safeguarded Dynamic Label Regression for Noisy Supervision," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 9103–9110, jul 2019.

[57] I. Misra, C. Lawrence Zitnick, M. Mitchell, and R. Girshick, "Seeing through the human reporting bias: Visual classifiers from noisy human-centric labels," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2930–2939, 2016.

[58] L. Jaehwan, Y. Donggeun, and K. Hyo-Eun, "Photometric transformer networks and label adjustment for breast density prediction," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pp. 0–0, 2019.

[59] B. Yuan, J. Chen, W. Zhang, H. S. Tai, and S. McMains, "Iterative cross learning on noisy labels," in *Proceedings - 2018 IEEE Winter Conference on Applications of Computer Vision, WACV 2018*, vol. 2018-Janua, pp. 757–765, 2018.

[60] A. Vahdat, "Toward robustness against label noise in training deep discriminative neural networks," in *Advances in Neural Information Processing Systems*, pp. 5596–5605, 2017.

[61] A. Veit, N. Alldrin, G. Chechik, I. Krasin, A. Gupta, and S. Belongie, "Learning from noisy large-scale datasets with minimal supervision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 839–847, 2017.

[62] M. Dehghani, A. Mehrjou, S. Gouws, J. Kamps, and B. Schölkopf, "Fidelity-weighted learning," in *International Conference on Learning Representations*, 2018.

[63] D. Tanaka, D. Ikami, T. Yamasaki, and K. Aizawa, "Joint optimization framework for learning with noisy labels," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5552–5560, 2018.

[64] K. Yi and J. Wu, "Probabilistic end-to-end noise correction for learning with noisy labels," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7017–7025, 2019.

[65] X. Liu, S. Li, M. Kan, S. Shan, and X. Chen, "Self-error-correcting convolutional neural network for learning with noisy labels," in *2017 12th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017)*, pp. 111–117, IEEE, 2017.

[66] S. Zheng, P. Wu, A. Goswami, M. Goswami, D. Metaxas, and C. Chen, "Error-bounded correction of noisy labels," in *International Conference on Machine Learning*, 2020.

[67] E. Arazo, D. Ortego, P. Albert, N. E. O'Connor, and K. McGuinness, "Unsupervised label noise modeling and loss correction," in *International Conference on Machine Learning*, 2019.

[68] J. Zhang, V. S. Sheng, T. Li, and X. Wu, "Improving crowdsourced label quality using noise correction," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 5, pp. 1675–1688, 2017.

[69] J. Han, P. Luo, and X. Wang, "Deep self-learning from noisy labels," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 5138–5147, 2019.

[70] J. Yao, J. Wang, I. W. Tsang, Y. Zhang, J. Sun, C. Zhang, and R. Zhang, "Deep learning from noisy image labels with quality embedding," *IEEE Transactions on Image Processing*, vol. 28, no. 4, pp. 1909–1922, 2018.

[71] T. Durand, N. Mehrasa, and G. Mori, "Learning a deep convnet for multi-label classification with partial labels," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 647–657, 2019.

[72] S. J. Delany, N. Segata, and B. Mac Namee, "Profiling instances in noise reduction," *Knowledge-Based Systems*, vol. 31, pp. 28–40, 2012.

[73] L. P. Garcia, J. A. Sáez, J. Luengo, A. C. Lorena, A. C. de Carvalho, and F. Herrera, "Using the one-vs-one decomposition to improve the performance of class noise filters via an aggregation strategy in multi-class classification problems," *Knowledge-Based Systems*, vol. 90, pp. 153–164, 2015.

[74] J. Luengo, S.-O. Shim, S. Alshomrani, A. Altalhi, and F. Herrera, "Cnc-nos: Class noise cleaning by ensemble filtering and noise scoring," *Knowledge-Based Systems*, vol. 140, pp. 27–49, 2018.

[75] C. G. Northcutt, T. Wu, and I. L. Chuang, "Learning with confident examples: Rank pruning for robust classification with noisy labels," in *Uncertainty in Artificial Intelligence - Proceedings of the 33rd Conference, UAI 2017*, may 2017.

[76] X. Wu, R. He, Z. Sun, and T. Tan, "A light CNN for deep face representation with noisy labels," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 11, pp. 2884–2896, 2018.

[77] J. Huang, L. Qu, R. Jia, and B. Zhao, "O2u-net: A simple noisy label detection approach for deep neural networks," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3326–3334, 2019.

[78] K. Sharma, P. Donmez, E. Luo, Y. Liu, and I. Z. Yalniz, "Noiserank: Unsupervised label noise reduction with dependence models," *arXiv preprint arXiv:2003.06729*, vol. 7, 2020.

[79] Y. Ding, L. Wang, D. Fan, and B. Gong, "A semi-supervised two-stage approach to learning from noisy labels," in *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1215–1224, IEEE, 2018.

[80] D. T. Nguyen, T.-P.-N. Ngo, Z. Lou, M. Klar, L. Beggel, and T. Brox, "Robust learning under label noise with iterative noise-filtering," *arXiv preprint arXiv:1906.00216*, 2019.

[81] 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*, 2020.

[82] Y. Yan, Z. Xu, I. W. Tsang, G. Long, and Y. Yang, "Robust semi-supervised learning through label aggregation," in *30th AAAI Conference on Artificial Intelligence, AAAI 2016*, pp. 2244–2250, 2016.

[83] J. Jiang, J. Ma, Z. Wang, C. Chen, and X. Liu, "Hyperspectral image classification in the presence of noisy labels," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 2, pp. 851–865, 2019.

[84] B. Han, I. W. Tsang, L. Chen, P. Y. Celina, and S.-F. Fung, "Progressive stochastic learning for noisy labels," *IEEE transactions on neural networks and learning systems*, no. 99, pp. 1–13, 2018.

[85] 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*, pp. 2304–2313, 2018.

[86] H.-S. Chang, E. Learned-Miller, and A. McCallum, "Active bias: Training more accurate neural networks by emphasizing high variance samples," in *Advances in Neural Information Processing Systems*, pp. 1002–1012, 2017.

[87] Y. Lyu and I. W. Tsang, "Curriculum loss: Robust learning and generalization against label corruption," in *International Conference on Learning Representations*, 2020.

[88] 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 *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 135–150, 2018.

[89] S. Reed, H. Lee, D. Anguelov, C. Szegedy, D. Erhan, and A. Rabinovich, "Training deep neural networks on noisy labels with bootstrapping," *arXiv preprint arXiv:1412.6596*, 2014.

[90] E. Malach and S. Shalev-Shwartz, "Decoupling" when to update" from" how to update"," in *Advances in Neural Information Processing Systems*, pp. 960–970, 2017.

[91] 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*, pp. 8527–8537, 2018.

[92] 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*, 2019.

[93] X. Wang, S. Wang, J. Wang, H. Shi, and T. Mei, "Co-mining: Deep face recognition with noisy labels," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 9358–9367, 2019.

[94] P. Chen, B. B. Liao, G. Chen, and S. Zhang, "Understanding and utilizing deep neural networks trained with noisy labels," in *International Conference on Machine Learning*, pp. 1062–1070, 2019.

[95] M. Ren, W. Zeng, B. Yang, and R. Urtasun, "Learning to reweight examples for robust deep learning," *arXiv preprint arXiv:1803.09050*, 2018.

[96] S. Jenni and P. Favaro, "Deep bilevel learning," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 618–633, 2018.

[97] J. Shu, Q. Xie, L. Yi, Q. Zhao, S. Zhou, Z. Xu, and D. Meng, "Meta-weight-net: Learning an explicit mapping for sample weighting," in *Advances in Neural Information Processing Systems*, pp. 1917–1928, 2019.

[98] Y. Wang, W. Liu, X. Ma, J. Bailey, H. Zha, L. Song, and S.-T. Xia, "Iterative learning with open-set noisy labels," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8688–8696, 2018.

[99] 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*, 2019.

[100] T. Liu and D. Tao, "Classification with noisy labels by importance reweighting," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 38, no. 3, pp. 447–461, 2015.

[101] R. Wang, T. Liu, and D. Tao, "Multiclass learning with partially corrupted labels," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 6, pp. 2568–2580, 2018.

[102] K.-H. Lee, X. He, L. Zhang, and L. Yang, "Cleannet: Transfer learning for scalable image classifier training with label noise," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5447–5456, 2018.

[103] O. Litany and D. Freedman, "Soseleto: A unified approach to transfer learning and training with noisy labels," in *International Conference on Learning Representations workshop on Learning from Limited Labeled Data*, 2018.

[104] W. Hu, Y. Huang, F. Zhang, and R. Li, "Noise-tolerant paradigm for training face recognition cnns," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 11887–11896, 2019.

[105] V. C. Raykar, S. Yu, L. H. Zhao, A. Jerebko, C. Florin, G. H. Valadez, L. Bogoni, and L. Moy, "Supervised learning from multiple experts: whom to trust when everyone lies a bit," in *Proceedings of the 26th Annual international conference on machine learning*, pp. 889–896, ACM, 2009.

[106] Y. Yan, R. Rosales, G. Fung, R. Subramanian, and J. Dy, "Learning from multiple annotators with varying expertise," *Machine Learning*, vol. 95, pp. 291–327, jun 2014.

[107] R. Tanno, A. Saeedi, S. Sankaranarayanan, D. C. Alexander, and N. Silberman, "Learning from noisy labels by regularized estimation of annotator confusion," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 11244–11253, 2019.

[108] F. Rodrigues and F. C. Pereira, "Deep learning from crowds," in *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, pp. 1611–1618, 2018.

[109] J. Whitehill, T.-f. Wu, J. Bergsma, J. R. Movellan, and P. L. Ruvolo, "Whose vote should count more: Optimal integration of labels from labelers of unknown expertise," in *Advances in neural information processing systems*, pp. 2035–2043, 2009.

[110] S. Branson, G. Van Horn, and P. Perona, "Lean crowdsourcing: Combining humans and machines in an online system," in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 6109–6118, 2017.

[111] H. Izadinia, B. C. Russell, A. Farhadi, M. D. Hoffman, and A. Hertzmann, "Deep classifiers from image tags in the wild," in *Proceedings of the 2015 Workshop on Community-Organized Multimodal Mining: Opportunities for Novel Solutions*, pp. 13–18, ACM, 2015.

[112] N. Manwani and P. Sastry, "Noise tolerance under risk minimization," *IEEE transactions on cybernetics*, vol. 43, no. 3, pp. 1146–1151, 2013.

[113] A. Ghosh, N. Manwani, and P. Sastry, "Making risk minimization tolerant to label noise," *Neurocomputing*, vol. 160, pp. 93–107, 2015.

[114] N. Charoenphakdee, J. Lee, and M. Sugiyama, "On symmetric losses for learning from corrupted labels," in *International Conference on Machine Learning*, 2019.

[115] P. L. Bartlett, M. I. Jordan, and J. D. McAuliffe, "Convexity, classification, and risk bounds," *Journal of the American Statistical Association*, vol. 101, no. 473, pp. 138–156, 2006.

[116] A. Ghosh, H. Kumar, and P. Sastry, "Robust loss functions under label noise for deep neural networks," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[117] X. Wang, Y. Hua, E. Kodirov, and N. M. Robertson, "Imae for noise-robust learning: Mean absolute error does not treat examples equally and gradient magnitude's variance matters," *arXiv preprint arXiv:1903.12141*, 2019.

[118] Z. Zhang and M. Sabuncu, "Generalized cross entropy loss for training deep neural networks with noisy labels," in *Advances in neural information processing systems*, pp. 8778–8788, 2018.

[119] Y. Wang, X. Ma, Z. Chen, Y. Luo, J. Yi, and J. Bailey, "Symmetric cross entropy for robust learning with noisy labels," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 322–330, 2019.

[120] N. Natarajan, I. S. Dhillon, P. K. Ravikumar, and A. Tewari, “Learning with noisy labels,” in *Advances in neural information processing systems*, pp. 1196–1204, 2013.

[121] V. Mnih and G. E. Hinton, “Learning to label aerial images from noisy data,” in *Proceedings of the 29th International conference on machine learning (ICML-12)*, pp. 567–574, 2012.

[122] Y. Xu, P. Cao, Y. Kong, and Y. Wang, “L_dmi: A novel information-theoretic loss function for training deep nets robust to label noise,” in *Advances in Neural Information Processing Systems*, pp. 6222–6233, 2019.

[123] G. Patrini, F. Nielsen, R. Nock, and M. Carioni, “Loss factorization, weakly supervised learning and label noise robustness,” in *International conference on machine learning*, pp. 708–717, 2016.

[124] B. Van Rooyen, A. Menon, and R. C. Williamson, “Learning with symmetric label noise: The importance of being unhinged,” in *Advances in Neural Information Processing Systems*, pp. 10–18, 2015.

[125] B. Han, I. W. Tsang, and L. Chen, “On the convergence of a family of robust losses for stochastic gradient descent,” in *Joint European conference on machine learning and knowledge discovery in databases*, pp. 665–680, Springer, 2016.

[126] L. P. Garcia, A. C. de Carvalho, and A. C. Lorena, “Noise detection in the meta-learning level,” *Neurocomputing*, vol. 176, pp. 14–25, 2016.

[127] B. Han, G. Niu, J. Yao, X. Yu, M. Xu, I. Tsang, and M. Sugiyama, “Pumpout: A meta approach for robustly training deep neural networks with noisy labels,” *arXiv preprint arXiv:1809.11008*, 2018.

[128] J. Li, Y. Wong, Q. Zhao, and M. S. Kankanhalli, “Learning to learn from noisy labeled data,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5051–5059, 2019.

[129] Y. Li, J. Yang, Y. Song, L. Cao, J. Luo, and L.-J. Li, “Learning from noisy labels with distillation,” in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1910–1918, 2017.

[130] M. Dehghani, A. Severyn, S. Rothe, and J. Kamps, "Learning to learn from weak supervision by full supervision," *arXiv preprint arXiv:1711.11383*, 2017.

[131] M. Dehghani, A. Severyn, S. Rothe, and J. Kamps, "Avoiding your teacher's mistakes: Training neural networks with controlled weak supervision," *arXiv preprint arXiv:1711.00313*, 2017.

[132] G. Algan and I. Ulusoy, "Meta soft label generation for noisy labels," in *Proceedings of the 25th International Conferance on Pattern Recognition, ICPR, pp. 7142–7148*, 2020.

[133] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[134] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[135] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," in *International Conference on Learning Representations*, 2018.

[136] G. Pereyra, G. Tucker, J. Chorowski, Ł. Kaiser, and G. Hinton, "Regularizing neural networks by penalizing confident output distributions," *arXiv preprint arXiv:1701.06548*, 2017.

[137] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

[138] D. Hendrycks, K. Lee, and M. Mazeika, "Using pre-training can improve model robustness and uncertainty," in *International Conference on Machine Learning*, pp. 2712–2721, PMLR, 2019.

[139] I. Jindal, M. Nokleby, and X. Chen, "Learning deep networks from noisy labels with dropout regularization," in *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pp. 967–972, IEEE, 2016.

[140] 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 Learning Representations*, 2018.

[141] S. Azadi, J. Feng, S. Jegelka, and T. Darrell, "Auxiliary image regularization for deep cnns with noisy labels," in *International Conference on Learning Representations*, 2016.

[142] X. Sun and H. Zhou, "An empirical comparison of two boosting algorithms on real data sets with artificial class noise," in *Advances in Information Technology and Education*, pp. 23–30, Springer, 2011.

[143] J. Cao, S. Kwong, and R. Wang, "A noise-detection based adaboost algorithm for mislabeled data," *Pattern Recognition*, vol. 45, no. 12, pp. 4451–4465, 2012.

[144] J. Bootkrajang and A. Kabán, "Boosting in the presence of label noise," in *Uncertainty in Artificial Intelligence - Proceedings of the 29th Conference, UAI 2013*, pp. 82–91, sep 2013.

[145] Q. Miao, Y. Cao, G. Xia, M. Gong, J. Liu, and J. Song, "Rboost: label noise-robust boosting algorithm based on a nonconvex loss function and the numerically stable base learners," *IEEE transactions on neural networks and learning systems*, vol. 27, no. 11, pp. 2216–2228, 2015.

[146] B. Sun, S. Chen, J. Wang, and H. Chen, "A robust multi-class adaboost algorithm for mislabeled noisy data," *Knowledge-Based Systems*, vol. 102, pp. 87–102, 2016.

[147] X. Yu, T. Liu, M. Gong, and D. Tao, "Learning with biased complementary labels," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 68–83, 2018.

[148] Y. Kim, J. Yim, J. Yun, and J. Kim, "Nlnl: Negative learning for noisy labels," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 101–110, 2019.

[149] Y. Xia, X. Cao, F. Wen, G. Hua, and J. Sun, “Learning discriminative reconstructions for unsupervised outlier removal,” in *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2015 Inter, pp. 1511–1519, 2015.

[150] Y. Duan and O. Wu, “Learning with Auxiliary Less-Noisy Labels,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 7, pp. 1716–1721, 2017.

[151] S. C. S. H. S. Lim and K. Brain, “Choicenet: Robust learning by revealing output correlations,” *arXiv preprint arXiv:1805.06431*, 2018.

[152] W. Zhang, Y. Wang, and Y. Qiao, “Metacleaner: Learning to hallucinate clean representations for noisy-labeled visual recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7373–7382, 2019.

[153] P. H. Seo, G. Kim, and B. Han, “Combinatorial inference against label noise,” in *Advances in Neural Information Processing Systems*, pp. 1171–1181, 2019.

[154] L. Niu, W. Li, and D. Xu, “Visual recognition by learning from web data: A weakly supervised domain generalization approach,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2774–2783, 2015.

[155] B. Zhuang, L. Liu, Y. Li, C. Shen, and I. Reid, “Attend in groups: a weakly-supervised deep learning framework for learning from web data,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1878–1887, 2017.

[156] B. Han, J. Yao, G. Niu, M. Zhou, I. Tsang, Y. Zhang, and M. Sugiyama, “Masking: A new perspective of noisy supervision,” in *Advances in Neural Information Processing Systems*, pp. 5836–5846, 2018.

[157] A. P. Dawid and A. M. Skene, “Maximum likelihood estimation of observer error-rates using the em algorithm,” *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, vol. 28, no. 1, pp. 20–28, 1979.

[158] J.-X. Zhong, N. Li, W. Kong, S. Liu, T. H. Li, and G. Li, "Graph convolutional label noise cleaner: Train a plug-and-play action classifier for anomaly detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1237–1246, 2019.

[159] C. Li, V. S. Sheng, L. Jiang, and H. Li, "Noise filtering to improve data and model quality for crowdsourcing," *Knowledge-Based Systems*, vol. 107, pp. 96–103, 2016.

[160] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proceedings of the 26th annual international conference on machine learning*, pp. 41–48, ACM, 2009.

[161] M. P. Kumar, B. Packer, and D. Koller, "Self-paced learning for latent variable models," in *Advances in Neural Information Processing Systems*, pp. 1189–1197, 2010.

[162] L. Jiang, D. Meng, S.-I. Yu, Z. Lan, S. Shan, and A. Hauptmann, "Self-paced learning with diversity," in *Advances in Neural Information Processing Systems*, pp. 2078–2086, 2014.

[163] H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek, "Loop: local outlier probabilities," in *Proceedings of the 18th ACM conference on Information and knowledge management*, pp. 1649–1652, ACM, 2009.

[164] J. Vuurens, A. P. de Vries, and C. Eickhoff, "How much spam can you take? an analysis of crowdsourcing results to increase accuracy," in *Proc. ACM SIGIR Workshop on Crowdsourcing for Information Retrieval (CIR'11)*, pp. 21–26, 2011.

[165] P. Wais, S. Lingamneni, D. Cook, J. Fennell, B. Goldenberg, D. Lubarov, D. Marin, and H. Simons, "Towards building a high-quality workforce with mechanical turk," *Proceedings of computational social science and the wisdom of crowds (NIPS)*, pp. 1–5, 2010.

[166] P. G. Ipeirotis, F. Provost, and J. Wang, "Quality management on amazon mechanical turk," in *Proceedings of the ACM SIGKDD workshop on human computation*, pp. 64–67, ACM, 2010.

[167] Y. Kong, “Dominantly truthful multi-task peer prediction with a constant number of tasks,” in *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 2398–2411, SIAM, 2020.

[168] A. Rasmus, M. Berglund, M. Honkala, H. Valpola, and T. Raiko, “Semi-supervised learning with ladder networks,” in *Advances in neural information processing systems*, pp. 3546–3554, 2015.

[169] 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*, pp. 1195–1204, 2017.

[170] M. E. Houle, “Dimensionality, discriminability, density and distance distributions,” in *2013 IEEE 13th International Conference on Data Mining Workshops*, pp. 468–473, IEEE, 2013.

[171] T. G. Dietterich, “An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization,” *Machine learning*, vol. 40, no. 2, pp. 139–157, 2000.

[172] D. I. Inouye, P. Ravikumar, P. Das, and A. Dutta, “Hyperparameter selection under localized label noise via corrupt validation,” in *NIPS Workshop*, 2017.

[173] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, “Imagenet large scale visual recognition challenge,” *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[174] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, “Inception-v4, inception-resnet and the impact of residual connections on learning,” *arXiv preprint arXiv:1602.07261*, 2016.

[175] G. Algan, I. Ulusoy, Ş. Gönül, B. Turgut, and B. Bakbak, “Deep learning from small amount of medical data with noisy labels: A meta-learning approach,” *arXiv preprint arXiv:2010.06939*, 2020.

[176] J. Cuadros and G. Bresnick, “Eyepacs: an adaptable telemedicine system for diabetic retinopathy screening,” *Journal of diabetes science and technology*, vol. 3, no. 3, pp. 509–516, 2009.

[177] F. Yu, A. Seff, Y. Zhang, S. Song, T. Funkhouser, and J. Xiao, "Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop," *arXiv preprint arXiv:1506.03365*, 2015.

[178] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, "Places: A 10 million image database for scene recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 6, pp. 1452–1464, 2017.

[179] T. Kaneko, Y. Ushiku, and T. Harada, "Label-noise robust generative adversarial networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2467–2476, 2019.

[180] K. K. Thekumparampil, A. Khetan, Z. Lin, and S. Oh, "Robustness of conditional GANs to noisy labels," in *Advances in Neural Information Processing Systems*, vol. 2018-Decem, pp. 10271–10282, 2018.

[181] E. Fonseca, M. Plakal, D. P. Ellis, F. Font, X. Favory, and X. Serra, "Learning Sound Event Classifiers from Web Audio with Noisy Labels," in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, vol. 2019-May, pp. 21–25, 2019.

Curriculum Vitae

Personal Information

Surname, Name: Algan, Görkem

Nationality: Turkish (TC)

Date and Place of Birth: 4 September 1989, Konya

Phone: +90 533 079 89 90

email: gorkemalgan@gmail.com

Education

Degree	Institution	Graduation
MS	Royal Institute of Technology, Stockholm	2014
MS	Eindhoven University of Technology, Eindhoven	2014
BS	METU Electrical-Electronics Engineering, Ankara	2012

Work Experience

Year	Place	Enrollment
2015-Present	ASELSAN, Ankara	Senior Computer Vision Engineer

Foreign Languages

Advanced English.

Publications

1. G. Algan and I. Ulusoy, “Image classification with deep learning in the presence of noisy labels: A survey,”Knowledge-Based Systems, vol. 215,p. 106771, 2021

2. G. Algan and I. Ulusoy, “Meta soft label generation for noisy labels” in Proceedings of the 25th International Conference on Pattern Recognition, ICPR, pp. 7142–7148, 2020