A Study of Robust Learning under Label Noise with Neural Networks

A Thesis
Submitted For the Degree of
Master of Technology (Research)
in the Faculty of Engineering

by

Deep Patel

Under the guidance of

P. S. Sastry

Department of Electrical Engineering
Indian Institute of Science
BANGALORE – 560 012

April 2021

TO

Nisha
without whom I wouldn't be here
and
I wouldn't be who I am

Acknowledgements

IISc has been a catalyst for change in my life. I have met some of the most honest and empathetic people here. In many ways I have learned how to think and feel because of these people. IISc has been a home for me. *The* home. And these people here have been my family. My *family*. Most of all, I have to thank Nisha. I am what I am and I am where I am because of you. It's you who has changed my life single-handedly. It's you who stood by me during those difficult times. I will always be grateful to you for all that you have done for me. In addition to meeting these amazing people, I also got to work with my amazing advisor, Prof. P.S. Sastry, who has been extremely supportive, kind, and caring throughout my journey here at IISc. You have been a friend and a mentor to me. I cannot thank you enough for the huge amounts of patience that you have had for me, especially for the last one year ever since the CoViD-19 pandemic began. Not only did you give me a place to stay, you stood by me even when I was failing to produce any research output. Your continued support and belief in me is what has led me to be able to produce this thesis work. I will always be grateful for the kindness that you have shown to me. Your ethical and honest way of doing research is something I will be carrying with myself for the rest of my life. I feel lucky to have been working with someone like you who has so much life and research experience.

I thank Vidyadhar and Safitha for bearing with me throughout the 2020 lockdown. I wouldn't be here at IISc if Prasanta Sir hadn't taken me in as a project assistant. It allowed me to get a better picture of life at IISc and the kind of research that is being carried out. All the SPIRE lab members were really supportive and helpful while preparing for admissions here. Thank you so much! I also want to thank Achutha for all the amazing discussions about machine learning research and research life in general. I really miss those interactions in C-mess. Big thanks to the C-Mess staff for cooking such amazing food and keeping me fed for all these years. It was like home food for me! Taking walks in this beautiful campus has always been therapeutic for me. I am really going to miss these walks! I am going to miss those ‘parties’ with Abhilash and Sagar featuring cocktails by Abhilash. I want to thank Lalit and Shivani for their company and support during this pandemic. It was a delightful experience to be part of NoteBook Drive (NBD) family and EMPATHS. Vishnu, I will miss those discussions about history of physics and politics and everything else. And how can I forget the *book-hopping* near Church Street whenever we could! Thank you, Avijith and Nisarg, for all the discussions about everything in the world. Thank you, Kartik and Shivam, for bearing with the numerous phone calls for no particular reasons. Without you, Maqbool Sir, I wouldn’t have discovered the joy of solving maths problems; especially, calculus.

Thank you, Shridhar, for the support that you have provided in the last few months. It’s partly because of this that I have finally reached this stage where I am writing the acknowledgments for my thesis. I am grateful to the institute administration for ensuring that life here at the campus was as close to normal as it could have been in spite of the raging pandemic. I wouldn’t be studying here if it were not for the tremendous support from my Foi and Kaka. Without you two, I wouldn’t have been able to pursue my research here. I can’t thank you two enough for this! And lastly, I am indebted to my father for providing me invaluable lessons on precisely what not to do. I have so many more people to thank that I am sure I have missed out on many names. Apologies to those whose names have been left out but know that I am grateful to you for all the love, kindness, and help you have showered me with throughout my life.

– Deep Patel

Declaration

I, **Deep Patel**, with SR No. **04-03-05-10-22-18-1-16151** hereby declare that the material presented in the thesis titled

A Study of Robust Learning under Label Noise with Neural Networks

represents original work carried out by me in the **Department of Electrical Engineering** at **Indian Institute of Science** during the years 2018-2021.

With my signature, I certify that:

- I have not manipulated any of the data or results.
- I have not committed any plagiarism of intellectual property.
- I have clearly indicated and referenced the contributions of others.
- I have explicitly acknowledged all collaborative research and discussions.
- I have understood that any false claim will result in severe disciplinary action.
- I have understood that the work may be screened for any form of academic misconduct.

Date: Student Signature

In my capacity as supervisor of the above-mentioned work, I certify that the above statements are true to the best of my knowledge, and I have carried out due diligence to ensure the originality of the thesis.

Advisor Name: Prof. P.S. Sastry Advisor Signature

Publications based on this Thesis

1. **D Patel**, P.S. Sastry, "Memorization in Deep Neural Networks: Does the Loss Function Matter?", *Accepted for* 25th Pacific-Asia Conference on Knowledge Discovery and Data Mining, New Delhi, India, 2021
2. **D Patel**, P.S. Sastry, "Adaptive Sample Selection for Robust Learning under Label Noise", *under review*

Abstract

Label noise is inevitable when employing supervised learning based algorithms in practice. In many applications involving neural networks one needs a large training set and the process of obtaining such labelled data (e.g., crowd sourcing, employing automatic web searches etc.) often lead to training set labels being noisy. In the context of neural networks, it is demonstrated that standard algorithms (such as minimizing empirical risk with cross entropy loss function) are susceptible to overfitting in the presence of noise. This thesis explores the problem of robust learning under label noise.

There are many approaches proposed for designing learning algorithms that are robust to label noise. We look at the sample reweighting methods wherein one tries to assign weights to different samples so that samples with noisy labels are assigned small or zero weights. This can be viewed as a kind of 'curriculum learning' wherein the clean (easy) samples are to be given more weightage than the corrupted (hard) samples. Based on such heuristics, we propose a simple, adaptive curriculum-based learning strategy called ***BA****tch* ***RE****weighting (BARE). The statistics of loss values of all samples in a mini-batch are used to decide which samples in each mini-batch would be allowed to update the weights. This yields an adaptive curriculum where the sample selection is naturally tied to current state of learning. Our algorithm does not need any clean validation data, needs no knowledge at all of the noise rates and also does not have any hyperparameters. We empirically demonstrate the effectiveness of our algorithm on benchmark data sets such as MNIST, CIFAR-10 and Clothing-1M, and show that it is much more efficient in terms of time and has as good or better robustness compared to other current algorithms based on sample reweighting.*

We next consider another aspect of the susceptibility of deep networks to label noise. It is shown recently that deep networks trained on data with random labels can memorize *the data in the sense of being able to drive the training error to zero. This phenomena of memorization is confirmed by multiple studies and it is seen that none of the standard regularization techniques can mitigate it. This depends on the kind of local minima that SGD can take the network to. Hence it could depend on the topography of the empirical risk that is minimized. Thus, the choice of loss function can be critical in determining this. However, none of the studies on memorization investigate the role of loss function. We present extensive empirical results to show that while standard loss functions like CCE and MSE result in memorization, symmetric loss functions such as RLL can resist such memorization to a good degree. We formally define what 'resisting memorization' means and then provide some theoretical justifications for the empirical results.*

Keywords

supervised learning, label noise, neural networks, curriculum learning, memorization

Contents

List of Figures

List of Symbols

$\mathbb{R}$	Set of real numbers
$\mathbb{R}_+$	Set of non-negative real numbers
$\mathbb{R}_{++}$	Set of positive real numbers
$\mathbb{Z}$	Set of integers
$\mathbb{R}^n$	Cartesian product of $\mathbb{R}$ with itself n times
$\mathbb{R}^n_+$	Cartesian product of $\mathbb{R}_+$ with itself n times
$\mathbf{1}$	Vector, of appropriate size, with all elements equal to 1
$\mathbf{0}$	Vector, of appropriate size, with all elements equal to 0
$\mathbf{e}_i$	Vector, of appropriate size, with i^{th} element equal to 1 and 0 for all other elements
$\mathbf{v} \geq 0$	Term wise inequality for vector $\mathbf{v} \in \mathbb{R}^n$
$[\mathbf{M}]_{ij}$	ij-th element of matrix $\mathbf{M}$
$\varnothing$	Empty set
h	Classifier function
$[K]$	$\{1, 2, \ldots, K\}$ where $K \in \mathbb{Z}_+$
∇f	Gradient of $f(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$, , *i.e.*, the i^{th} component of ∇f is $\frac{\partial f}{\partial \mathbf{x}_i}$

List of Abbreviations

SVM	Support Vector Machine
DT	Decision Tree
SLN	Symmetric Label Noise
CCLN	Class-Conditional Label Noise
NULN	Non-Uniform Label Noise
CCE	Categorical Cross-Entropy Loss
MSE	Mean Squared Error Loss
MAE	Mean Absolute Error Loss
NN	Neural Networks
CoT	Co-Teaching
CoT+	Co-Teaching+
MR	Meta-Ren
MN	Meta-Net
CL	Curriculum Loss
BARE	BAtch REweighting
RLL	Robust Log Loss

Chapter 1

Introduction

The supervised classification problem in Machine Learning is one of the most widely studied problems with a plethora of applications. Typically, one has access to a labelled training set that is used to train a model (which is referred to as a classifier). This is also called *supervised learning* as one has some patterns along with corresponding labels that indicate which category the pattern belongs to. In recent years the neural network models have been spectacularly successful in many classification applications. However, these models need a large training set of labelled patterns. While creating such large scale datasets, often, there are labelling errors owing to automated labelling processes, crowdsourced annotations, human errors, etc. Hence, in these cases, the training set contains labelling errors. Thus the problem of robust learning under label noise is very relevant and needs to be studied in detail. This thesis presents a study on some aspects of robust learning under label noise. In this chapter we provide a general introduction to supervised learning and describe the problem of robust learning under label noise.

1.1 Supervised Learning

In supervised machine learning, we are given a training set of i.i.d. samples $S = \{(\mathbf{x}_i, y_i),\ i = 1, \cdots, m\}$, $\mathbf{x}_i \in \mathcal{X}, y_i \in \mathcal{Y}$, drawn from some underlying distribution, $\mathcal{D}$ on $\mathcal{X} \times \mathcal{Y}$. The goal is to learn a function, $f : \mathcal{X} \to \mathcal{Y}$. In particular, consider the case of multi-class classification problem, i.e., $\mathcal{Y} = \{1, 2, \ldots, K\}$ where $K \in \mathbb{Z}_+$ denotes the number of cateogries or classes[1]. Here, $\mathbf{x}_i$'s ($\in \mathcal{X} \subseteq \mathbb{R}^n$) are referred to as patterns or features and y_i's as labels. What's often done for classification problems is that one first learns a score function, $f : \mathcal{X} \rightarrow \mathcal{M}$ where $\mathcal{M} \subseteq \mathbb{R}^K$. Then, a mapping function is used to predict a label for the pattern using the corresponding score. So, the classifier function that we learn can be written as $h(\cdot) = \mathit{pred} \circ f(\cdot)$ where 'pred' is the mapping function used to convert scores to labels. For instance, pred($\mathbf{a}$) could be the index corresponding to the maximum component of $\mathbf{a} \in \mathcal{M}$.

1.2 Risk Minimization Framework

Many of the (often-used) supervised learning methods such as Support Vector Machines, logistic regression, perceptron, AdaBoost, neural networks etc. can be formulated as *risk minimization* problems. The *risk* is defined with respect to a so called *loss function*. A loss function, $\mathcal{L}$, maps scores ($\mathcal{M} \subseteq \mathbb{R}^K$) and label values ($\mathcal{Y}$) to non-negative real values, i.e., $\mathcal{L} : \mathcal{M} \times \mathcal{Y} \rightarrow \mathbb{R}_+$. The loss function is user-defined and is to be chosen such that it suitably captures the objectives of the learning process. *Risk* is defined to be the expected value of the loss function, $\mathcal{L}$, w.r.t. the underlying distribution of the data, $\mathcal{D}$. We will refer to the *risk* with respect to a loss function $\mathcal{L}$ as $\mathcal{L}$-risk from now on and it can be formally written as follows:

$$R_{\mathcal{L}}(f) = \mathbb{E}_{\mathcal{D}}[\mathcal{L}(f(\mathbf{x}), y_{\mathbf{x}})] \tag{1.1}$$

If we are using risk minimization framework, then the global minimizer of risk (defined in Equation 1.1) is the optimal, and hence, the desired classifier. A global minimizer of this risk will be denoted by f^*.

$$f^* = \arg \min_{f \in \mathcal{F}} R_{\mathcal{L}}(f) = \arg \min_{f \in \mathcal{F}} \mathbb{E}_{\mathcal{D}}[\mathcal{L}(f(\mathbf{x}), y_{\mathbf{x}})] \tag{1.2}$$

where $\mathcal{F}$ is a hypothesis space over which we learn a classifier function.

As an illustration, let's look at risk with respect to 0–1 loss function. This loss function is defined as:

[1]we denote $\{1, 2, \ldots, K\}$ as $[K]$ from now on

$$\mathcal{L}(f(\mathbf{x}), y_{\mathbf{x}}) = \begin{cases} 0 & (pred \circ f(\mathbf{x})) = y_{\mathbf{x}} \\ 1 & (pred \circ f(\mathbf{x})) \neq y_{\mathbf{x}} \end{cases} \tag{1.3}$$

The risk under 0–1 loss function can be written as follows:

$$\begin{aligned} R_{0-1}(f) &= \mathbb{E}_{\mathcal{D}}[\mathcal{L}_{0-1}(f(\mathbf{x}), y_{\mathbf{x}})] = \mathbb{E}_{\mathcal{D}}[\mathbb{I}_{(pred \circ f(\mathbf{x})) \neq y_{\mathbf{x}}}] \\ &= \text{Prob}_{\mathcal{D}}[(pred \circ f(\mathbf{x})) \neq y_{\mathbf{x}}] \end{aligned} \tag{1.4}$$

Thus the minimizer of risk here would be the minimizer of probability of misclassification.

Since 0-1 loss function is non-differentiable, it's difficult to optimize the risk under this loss function. To circumvent this, surrogate losses of 0-1 loss function are used as optimization is easier in that case. Some examples of surrogate losses for 0-1 loss function in case of binary classification are sigmoid loss, mean squared error (MSE) loss, and hinge loss.

Since one doesn't have knowledge about the true, underlying data distribution, $\mathcal{D}$, in practice, we minimize the empirical risk using samples drawn from $\mathcal{D}$, i.e. the training set, S. The empirical risk over this training set, S, with respect to a loss function, $\mathcal{L}$ can be defined as:

$$\hat{R}_{\mathcal{L}}(f) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(f(\mathbf{x}_i), y_i) \tag{1.5}$$

The empirical risk, $\hat{R}_{\mathcal{L}}(f)$, is a good approximation of the risk, $R_{\mathcal{L}}(f)$, because, by law of large numbers, $\hat{R}_{\mathcal{L}}(f) \to R_{\mathcal{L}}(f)$ as $m \to \infty$. When this convergence is uniform over the family of functions over which we search for the minimizer, one can show that minimizer of empirical risk would also be (a good approximation of) minimizer of risk. So, most learning algorithms, after choosing a suitable loss function, minimize the empirical risk to learn a classifier function.

1.3 Learning under Label Noise

1.3.1 Why is it important?

For learning classifiers there are myriad algorithms available such as Support Vector Machines (SVMs), Decision Trees (DTs), AdaBoost, Neural Networks, etc. However, there is a caveat. All these learning algorithms have been formulated under the assumption that the training data comes from the same distribution ($\mathcal{D}$) on which we want the learnt classifier to perform well. In practice, however, one may not always have access to data from the true underlying distribution (or the distribution that we are interested in learning). Specifically, in this thesis, we are interested in the situations where the labels for the training set patterns do not conform to the desired underlying distribution. This can happen, for example, when for large scale image datasets, labels are obtained via crowd-sourcing (many of the labellers maybe non-experts) or web-crawlers (search engine queries may be inadequate). Subjective biases too can create erroneous labels. In such cases we say that the training data has label noise.

This problem of learning under label noise is especially pertinent in the current times given the widespread use of neural networks and large-scale datasets for various applications such as image recognition, speech recognition, etc.

1.3.2 Problem Formulation

We consider a K-class classification problem with $\mathcal{X}$ as the feature/pattern space. We take $\mathcal{Y} = \{0,1\}^K$ as the label space. We assume all labels are one-hot vectors and denote by $\mathbf{e}_k$ the one-hot vector corresponding to class k. Let $S = \{(\mathbf{x}_i, y_i^{cl}),\ i = 1, 2, \cdots, m\}$ be the i.i.d. samples drawn according to a distribution $\mathcal{D}$ on $\mathcal{X} \times \mathcal{Y}$. (We think of the labels here as 'clean' and hence the superscript). Let us assume we are interested in learning a classifier that does well on a test set drawn according to $\mathcal{D}$. We can do so if we are given S as the training set. However, we do not have access to this training set and what we have is a training set $S_\eta = \{(\mathbf{x}_i, y_i),\ i = 1, 2, \cdots, m\}$ drawn according to a distribution $\mathcal{D}_\eta$. The $y_{\mathbf{x}}$ here are the 'corrupted' or 'noisy' labels and they are random variables related to the 'clean' labels, $y_{\mathbf{x}}^{cl}$, through a noise model which can be specified through the following conditional probabilities:

$$\eta_{\mathbf{x},ij} = P(y_{\mathbf{x}} = j \mid \mathbf{x}, y_{\mathbf{x}}^{cl} = i), \quad i, j \in [K] \tag{1.6}$$

These conditional probabilities are called noise rates. Since these are conditional probabilities, we have $\sum_{j\in[K]} \eta_{\mathbf{x},ij} = 1, \ \forall\ i \in [K], \ \forall\ \mathbf{x}$. The training data avaialable, S_η, is drawn according to $\mathcal{D}_\eta$ whereas the clean data from $\mathcal{D}$ is not available. Note that $\mathcal{D}_\eta$ is determined from $\mathcal{D}$ and the conditional probabilities given by eq.(1.6). What we want is to be able to learn a classifier, f, that performs well on clean data, that is, data drawn from $\mathcal{D}$. (We are not interested in the performance of f on data from $\mathcal{D}_\eta$). Before we go ahead and mention different strategies to combat label noise, we categorize label noise into various types.

- Symmetric/Uniform label noise (SLN):Under such noise, the labels are uniformly flipped to any one of the remaining classes. This type of noise is characterized by the noise rates satisfying the following:

$$\begin{aligned} \eta_{\mathbf{x},ij} &= \frac{\eta}{K-1}\ \forall\ i, j \in [K], j \neq i,\ \forall \mathbf{x} \quad (1.7) \\ \eta_{\mathbf{x},ii} &= 1 - \eta \quad (1.8) \end{aligned}$$

  Stated in words, the noise rates $\eta_{\mathbf{x},ij}$ are dependent neither on the feature vectors ($\mathbf{x}$'s) nor the 'true' or 'clean' labels ($y_{\mathbf{x}}^{cl}$'s). Under symmetric noise we can think of S_η as derived from S as follows: for each sample in S we retain the same label with probability $1-\eta$ and with probability η the label is changed; when it is changed, it is equally likely to be any of the other labels. This noise model would be useful, e.g., in cases of random errors in crowd sourced data.

- Class-conditional label noise (CCLN): This is, relative to symmetric label noise, a more realistic model of label noise. As per this model, the noise rates can depend on the 'true' labels ($y_{\mathbf{x}}^{cl}$'s) but not on the feature vector, $\mathbf{x}$:

$$\eta_{\mathbf{x},ij} = \eta_{ij},\ \forall \mathbf{x} \tag{1.9}$$

Based on these probabilities, we can now define the noise rate matrix, $\mathbf{N}$, where $[\mathbf{N}]_{ij} = \eta_{\mathbf{x},ij} = P(y_{\mathbf{x}} = j \mid \mathbf{x}, y_{\mathbf{x}}^{cl} = i) = P(y_{\mathbf{x}} = j \mid y_{\mathbf{x}}^{cl} = i) = \eta_{ij}$, which completely specifies the noise model. This noise model is useful, for example, when different pairs of classes have different probabilities of being mistaken for each other. For instance, in MNIST dataset which contains images of hand-written digits, there could be confusion between digits 2 and 7, 3 and 8, 1 and 7, and so on.

- Non-Uniform label noise (NULN): This is the most general case of label noise. As per this model, the noise rates are dependent on both the feature vectors and the true labels.

The problem of learning in presence of label noise has a long history. For example, finite hypothesis spaces were proven to be PAC-learnable under symmetric label noise in the 1980's [3,106]. In the last decade or so, the problem of learning under label noise attracted a lot of attention because many studies have pointed out that the classifier learning methods get adversely affected if the training set has label noise. Long et al. [67] show that many boosting algorithms which essentially optimize a convex potential function perform poorly in presence of label noise. This susceptibility of boosting algorithms to symmetric label noise is empirically demonstrated by many other studies [21,51,75,76,84]. Similarly many standard algorithms such as naive Bayes, k-Nearest Neighbours (kNNs), SVMs, and logistic regression perform poorly in presence of symmetric label noise [10, 12, 83, 87, 103, 126]. Quinlan [91] empirically demonstrated for discrete-valued features that DTs perform poorly in presence of symmetric label noise. Ensemble methods fail as well in the presence of label noise [2]. Whence it's clear that these standard algorithms are not able to learn "good" classifiers when there's label noise. In the context of neural networks, [6, 123] demonstrate that neural networks are able to memorize the entire training dataset for any amount of label noise. That is, given training set with random labels, these neural networks can learn to exactly reproduce the random labels for training set patterns. This raises serious questions regarding the generalization capability of neural networks trained with data having label noise. Hence, it's pertinent to understand what algorithmic modifications or new methodologies aid in combating this problem of label noise while learning the classifier functions.

1.3.3 Robust Risk Minimization

Risk Minimization framework provides a language to pose the problem of learning under label noise in a way that helps us to understand many of the important theoretical issues for combating label noise [30].

Given that there is label noise, we will have access only to the noisy training set, S_η, whose samples are drawn according to the distribution, $\mathcal{D}_\eta$. However, we want to learn a classifier from this training data such that it performs well on data from the true, underlying distribution, $\mathcal{D}$. By robust risk minimization we mean a risk minimization strategy that can guarantee this under some conditions. There are two approaches for such robust risk minimization: i.) loss correction, and ii.) inherently robust loss functions.

Loss Correction

Loss Correction method involves modification of loss function such that the minimizer of risk under this modified loss with respect to noisy distribution, $\mathcal{D}_\eta$, would be same as minimizer of risk with the original loss function under the clean distribution, $\mathcal{D}$. Suppose we are interested in finding the minimizer of risk with respect to $\mathcal{L}$ and the clean data distribution. Then the idea of loss correction is to find a loss function $\bar{\mathcal{L}}$ that satisfies the following.

$$\begin{aligned} \mathbb{E}_{\mathcal{D}}[\mathcal{L}(f(\mathbf{x}), y_{\mathbf{x}}^{cl})] &= \quad \mathbb{E}_{\mathcal{D}_\eta}[\bar{\mathcal{L}}(f(\mathbf{x}), y_{\mathbf{x}})] \\ \text{which is same as} & \\ \mathbb{E}_{\mathbf{x}}\mathbb{E}_{y_{\mathbf{x}}^{cl}|\mathbf{x}}[\mathcal{L}(f(\mathbf{x}), y_{\mathbf{x}}^{cl})] &= \mathbb{E}_{\mathbf{x}}\mathbb{E}_{y_x|\mathbf{x}}[\bar{\mathcal{L}}(f(\mathbf{x}), y_{\mathbf{x}})] \end{aligned} \tag{1.10}$$

This was first proposed in [82] for binary classification to tackle symmetric label noise. Studies such as [43, 86] extend this method to multi-class classification case for robustness against symmetric and class-conditional label noise. It's clear from Equation 1.10 that a minimizer, of $\bar{\mathcal{L}}$-risk under noisy distribution (RHS) is also the minimizer of $\mathcal{L}$-risk under clean distribution (LHS). If one uses a special class of loss functions called proper composite losses [93], it can be guaranteed that the minimizer of RHS and LHS are the same without requiring Equation 1.10 to hold true.

The design of $\bar{\mathcal{L}}$ to satisfy Equation 1.10 needs knowledge of the noise rates. Thus, one caveat in this method of loss correction is that we need to know the noise rates beforehand, or be able to estimate them. In practice one does not know the noise rates and estimating them reliably from the data is difficult.

Inherently Robust Losses

Robust risk minimization can also be achieved by designing loss functions that are inherently robust. Risk minimization with loss function, $\mathcal{L}$, is defined to be robust under label noise if [73]:

$$\text{Prob}_{\mathcal{D}}(\{pred \circ f^*(\mathbf{x}) = y_{\mathbf{x}}\}) = \text{Prob}_{\mathcal{D}}(\{pred \circ f^*_{\eta}(\mathbf{x}) = y_{\mathbf{x}}\}) \tag{1.11}$$

where

$$\begin{aligned} f^*(\mathbf{x}) &= \arg\min_{f \in \mathcal{F}} R_{\mathcal{L}}(f) = \arg\min_{f \in \mathcal{F}} \mathbb{E}_{\mathcal{D}}[\mathcal{L}(f(\mathbf{x}), y^{cl}_{\mathbf{x}})] \\ f^*_{\eta}(\mathbf{x}) &= \arg\min_{f \in \mathcal{F}} R^{\eta}_{\mathcal{L}}(f) = \arg\min_{f \in \mathcal{F}} \mathbb{E}_{\mathcal{D}_{\eta}}[\mathcal{L}(f(\mathbf{x}), y_{\mathbf{x}})] \end{aligned}$$

When the above holds we say loss function $\mathcal{L}$ is robust. For binary classification, [73] show that 0–1 loss function and MSE are robust to symmetric label noise. And that 0–1 loss function is robust to non-uniform label noise as well if true Bayes risk is zero, i.e. $R_{\mathcal{L}}(f^*) = 0$. These results are extended in [30] to the case of multi-class classification by proving sufficient conditions for a special class of loss functions called *symmetric losses*. The results are independent of the data distribution and the hypothesis class, $\mathcal{F}$, over which a classifier is learnt.

A loss, $\mathcal{L}$, is called symmetric if $\exists\ C \in \mathbb{R}_{++}$ such that it satisfies the following condition:

$$\sum_{i=1}^{K} \mathcal{L}(f(\mathbf{x}), i) = C \quad \forall \mathbf{x}\, \forall f, \tag{1.12}$$

(Here we are taking $\mathcal{Y} = [K]$). We note here that, since we assume loss to be non-negative, from the above definition it follows that a symmetric loss function is bounded in the following sense:

$$\mathcal{L}(f(\mathbf{x}), j) \leq C,\ \forall f,\ \forall \mathbf{x},\ \forall j \in \mathcal{Y} \quad \text{(when } \mathcal{L} \text{ is a symmetric loss)} \tag{1.13}$$

It can be seen that 0–1 is a symmetric loss whereas MSE is not. Unlike the mean square error, the mean absolute error (MAE) is a loss that is symmetric. It is shown in [30] that if a loss function is symmetric, then risk minimization is robust for symmetric label noise if $\eta < \frac{K-1}{K}$ and to (a special type of) non-uniform label noise if $\eta_{\mathbf{x}} < \frac{K-1}{K}$ and $R_{\mathcal{L}}(f^*) = 0$. These results are based on the idea of robustness as defined in Equation 1.11. There's another notion of robustness [92] wherein the relative ranks of risk remains unchanged under noisy and clean distributions as seen in Equation 1.14:

$$R_{\mathcal{L}}(f_1) < R_{\mathcal{L}}(f_2) \Longleftrightarrow R^{\eta}_{\mathcal{L}}(f_1) < R^{\eta}_{\mathcal{L}}(f_2) \;\; \forall \, f_1, \; f_2 \; \in \mathcal{F} \tag{1.14}$$

This is a stronger notion of robustness as any local minimizer of $R_{\mathcal{L}}$ is a local minimizer of $R^{\eta}_{\mathcal{L}}$.

Symmetric loss functions offer many advantages for robust learning. One does not need to know (or estimate) noise rates or assume availability of clean validation data. And there is only a minimal change in the training algorithm, i.e. changing the loss function. However, the optimization problem of minimizing risk may be more difficult with some symmetric losses (e.g., 0–1 loss). Also, it may be noted that symmetry of a loss function is only a sufficient condition for robustness. In practice, we minimize empirical risk. So, if consistency of risk minimization holds true and if risk minimizer is robust, then empirical risk minimizer will also be robust given that we have enough number of noisy samples.

Apart from these robust risk minimization methods, there are many other algorithmic approaches for learning under label noise.

1.4 Approaches for tackling label noise

In this section we briefly mention some categories under which we can broadly categorize all the approaches for tackling label noise proposed in the literature so far. We discuss them in more detail in the next chapter. These different approaches are as follows:

- *Label filtering* methods [12, 18, 39, 52, 53, 119] attempt at identifying the samples that are likely to have incorrect labels. This is one of the oldest approaches and mainly incorporates removal of outliers based on heuristics sometimes supported by *robust statistics* theory.

- *Label cleaning* methods [102] attempt at identifying and correcting the potentially incorrect labels through joint optimization of labels and network weights [104,121] or treating true labels as latent variables and using EM-style algorithms to infer them [118]. Various heuristics such as using ensemble of classifiers [134] or entropy of the softmax output of a neural network [102] are also used to identify the most-likely-correct label.

- *Loss correction* methods [43,82,86,110] suitably modify loss function (or posterior probabilities) to correct for the effects of label noise on risk minimization such that minimizers of risk under the noisy distribution ($\mathcal{D}_\eta$) are same as those of risk under the clean distribution ($\mathcal{D}$); however, they need to know (or estimate) the noise rates.

- *Robust Loss Function* based methods devise loss functions which are inherently robust (Equation 1.11) thereby enabling robust risk minimization in presence of label noise [14,30,57,69,73,109,128].

- *Regularization* methods devise regularizers that penalize the network parameters if the network starts overfitting to noisy data and help the network to learn from clean data [4,61,63,70,79,92,124] instead.

- *Sample Reweighting* methods are one of the popular strategies for regularization to tackle the noisy data. We discuss this separately from regularization methods in this thesis because this thesis presents a new algorithm for sample reweighting. The idea in this approach is to optimize over a weighted loss wherein each sample's weight is adjusted such that more weightage is given to the data that is more likely to be 'clean' and overfitting to noisy data is reduced. There are several works that propose such a scheme [37,50,68,72,94,101,120,122].

1.5 Thesis Organization

We briefly describe the contributions of this thesis and its organization in this section.

Having formally stated the problem of learning under label noise and what it means to have robust risk minimization, in Chapter 2, we do an extensive review of the literature on robust learning under label noise. We provide a categorization for the existing methods and briely describe the existing methods under these categories.

In Chapter 3, we consider one of the popular methods for tackling label noise – *sample reweighting* – which consists of assigning binary or real-valued weights to different training samples and then minimizing the weighted loss to reduce the effect of samples with corrupted labels on the learning of neural network parameters. We propose a simple, adaptive (binary) sample reweighting strategy called **BA**tch **RE**weighting (BARE) that utilizes batch statistics of loss values in a given mini-batch for sample reweighting thereby tying this reweighting to the current state of learning. Unlike existing methods, the proposed algorithm does not need any clean validation data, needs no knowledge at all of the noise rates and also does not have any hyperparameters. We empirically demonstrate the effectiveness of our algorithm on benchmark datasets – MNIST and CIFAR-10 and Clothing-1M – and show that it is much more efficient in terms of time and has as good or better robustness compared to other algorithms for different types of label noise and noise rates. These results demonstrate that relying on current state of learning alone can help mitigate overfitting to noisy data and provide robust learning under label noise.

In Chapter 4, we analyze another susceptibility of deep neural networks to label noise, viz. 'memorization'. This phenomenon of 'memorization' – driving training set error to zero (or close to zero) – is poorly understood. This 'memorization' depends on the kind of local minima that SGD can take the network to which in turn depends on the topography of the empirical risk that is minimized. Hence, the choice of loss function can be critical in determining this. However, none of the studies on memorization investigate this. We perform an empirical study – on benchmark data sets MNIST and CIFAR-10 – and show that choice of loss function can affect memorization. We show that a special class of loss functions, namely, symmetric losses, can resist memorization to a good degree. We formally define what 'resisting memorization' means and provide some theoretical justifications for the empirical results.

In Chapter 5, we summarize the contributions of this thesis and discuss potential future directions for better robust learning under label noise with neural networks.

Chapter 2

Learning under Label Noise - A Review

As mentioned in the previous chapter, there have been many algorithms suggested for mitigating the adverse effects of label noise while learning a classifier. Many of the algorithms are heuristic in the sense that they do not guarantee robust learning. However, many algorithms are also quite effective in learning good classifiers when training set is noisily labelled.

In this chapter, we present a detailed review of methods for learning under label noise. For the purposes of this review, we categorize the approaches for learning under label noise into 6 categories: Label Filtering, Label Cleaning, Robust Loss Functions, Loss Correction, Regularization, and Sample Reweighting. These are discussed in the next six sections of this chapter.

2.1 Label Filtering

This is one of the oldest approaches adopted by the community. The idea is to identify and filter out or remove the *noisy samples* from the training set and then retrain the model with the remaining samples in the dataset. These *noisy samples* can be viewed as *outliers* in the sense that they represent *'a case that does not follow the same model as the rest of the data"* [113]. However, here it is not the feature vector that is an outlier because all $\mathbf{x}_i$ in the training data come from the same underlying distribution. It is pairing of $\mathbf{x}_i$ with a label (which is a categorical variable) that makes it an 'outlier' and the problem of dealing with such outliers is not typically addressed in the classical robust statistics literature [52]. Label filtering does provide one way to deal with this issue of such outliers. The difficulty in this approach is to find effective methods of distinguishing hard-to-learn but informative samples from the noisy ones [34].

Many label filtering algorithms are motivated by outlier detection. An outlier detecting algorithm is proposed in [119] by modifying the hinge loss to make SVM robust. It is empirically observed that the size of the Decision Trees (DTs) increases in the presence of label noise [91]. To help DTs become robust, [52] proposes a pruning method which effectively removes uninformative samples or outliers. After having removed the outliers, a new DT is learned with the cleaned-up data set.

Since *noisy samples* would have 'wrong' labels, a good heuristic is that any classifier learning method would find these examples difficult to learn. Thus, if we train many different classifiers on the data, then most of them may be classifying the noisy samples incorrectly and this can be used for detecting training data with wrong labels. This idea of detecting noisy samples via a majority-based decision of ensemble of classifiers (which are trained on subsets of training set) is referred to as classification filtering. One of the earliest such algorithms is proposed in [12] where an ensemble of classifiers is trained with n-fold cross-validation subsets of training data. A sample is declared noisy and discarded if majority of classifiers misclassify it. With a similar idea and with n-fold cross-validation subsets, [53] proposes a modified boosting approach wherein the weight of a sample in every iteration is increased if it's misclassified in the previous training step; once this weight exceeds a pre-defined threshold, it is deemed an outlier and discarded.

Another simple heuristic to detect wrong labels is the following: *all the points closer to the centroid of any other class than its own is probably an outlier*. This is used in [18] to propose a k-Nearest Neighbour (kNN) search to identify if a sample's label is different from the majority class of its neighbours and discard it in such a case. Another heuristic algorithm is proposed in [39] to obtain large-margin *core set* [7] which is a subset of training data for which the maximum margin hyperplane learnt from it is a good approximation of that learnt from the entire training data. These heuristics are seen to work well only if the noisy samples were near the class boundaries.

2.2 Label Cleaning

This is also one of the early approaches adopted by the research community. The idea is to clean the noisy samples from the training data set and thus learn a model with the cleaned-up training data set. The difficulty of distinguishing the hard-to-learn examples from noisily-labelled examples is present in this approach as well. One can use any of the label filtering approaches to identify the wrongly labelled data and then use some heuristics to clean the labels. This is useful because if we discard all samples detected as noisy, we may end up with significantly reduced training set. Many heuristics are proposed to infer the true label for a nosiy data sample. It is also possible to clean labels without explicitly identifying wrongly labelled data beforehand. An interesting and widely used approach is to treat the unknown but 'true' labels as latent variables and treat the given noisy training sample as incomplete data in an Expectation-Maximization (EM) style algorithm. One needs a noise model to link the observed data and the latent variables. Here the E-step would amount to label cleaning and the M-step corresponds to network parameter updation after label cleaning. Another approach is to use a joint optimization framework for learning the network parameters and clean labels together.

One of the heuristics for inferring true label for a noisy data sample is that most of the neighbours of a data point should be of the same class. This is used in [28] which proposes a kNN to assign a given sample the label of majority class among its k nearest neighbours. This heuristic does not work well near class boundaries unless there is a large margin between classes. To deal with label noise in case of online learning where data is available only in chunks, based on analysis of error rates of linearly combined neural classifiers done by [105], [134] proposed a greedy algorithm to learn an ensemble of classifiers trained on previous k chunks of data to reduce the classifier ensemble's error rate by minimizing (maximizing) the error-rate variance[1] of the classifier ensemble on the cleaned-up (noisy) parts of the most recent chunk of data. This process gives the weight-factor for each of the ensemble's classifiers which is then used to clean up the current data chunk by weighted-majority voting.

[1]This idea is referred to as Maximum Variance Margin (MVM) principle.

There are many algorithms that use the EM framework to clean labels. A label cleaning algorithm for binary classification under label noise is proposed in [80] by treating true labels as latent variables. The posterior probabilities for these latent, true labels are inferred with the help of an EM algorithm. They propose modifiying the gradient for CCE loss to make the network prediction close to posterior probability of the latent true label. An algorithm to infer the true labels and estimate noise rate matrix via a probabilistic graphical model for label noise is proposed in [118]. EM algorithm is used to infer the true labels which are then used for supervision by the neural network for the classification task.

As mentioned earlier, another method of label cleaning is to jointly learn the network parameters as well as the training set labels. A label cleaning network (with linear activations) in conjunction with the usual neural networks for classifier training was proposed in [108] and the learning algorithm jointly optimizes over network parameters and training labels. For the label cleaning network to be effective, however, the algorithm needs a small, clean validation set. They use the Mean Absolute Error (MAE) as the loss function. A joint optimization framework to learn both the label distribution (to clean the labels) and network parameters is also proposed in [104, 121]. These algorithms do not require extra clean data. In addition to this, entropy of network's posterior probabilities and cross-entropy between label distribution and network's posterior probabilities are used as regularizers during the training. A Markov Chain Sampling (MCS) based framework is proposed in [129] wherein the states of the Markov Chain (MC) are all possible subsets of the noisy training data. A set of classifiers are trained on different subsets of the training data. Authors show that, under the intuition and assumption that the noisy-labelled examples are noisily-labelled because of inconsistent 'mistakes' and hence do not come from the same distributions, it is more likely to end up in a "good" state (i.e. state majorly containing clean examples) as opposed to a "bad" state (i.e. state majorly containing noisy examples) in the MC. Based on this, authors propose to calculate the "expected state" which essentially indicates the probability that a sample is clean. This "expected state" can be used to filter, clean, or serve as sample-weights for the samples.

An algorithm based on consistency of history of label predictions is proposed in [102] for label cleaning. Here the fractions of times different labels predicted for a sample over the last q iterations is taken as a probability distribution over labels and a sample with low entropy of this distribution is treated as a clean sample. The labels of samples are cleaned by assigning the most frequent label prediction over the last-q times. Another method termed negative learning is proposed in [54] wherein the network is fed 'complementary' labels – classes which the sample does NOT belong to. This results in a low chance of selecting a true label as a 'complementary' label thereby reducing the risk of providing incorrect information to the algorithm. This is used to filter out clean and noisy training data after which we are to iteratively pick the highly confident samples for label cleaning. A meta-learning based method is proposed in [130]. A label correction network is jointly learned along with the classifier via meta-learning here. Zheng et al [131] provide an explanation for why neural networks trained with noisy data are helpful in identifying noisy data so that it can be filtered out or label-corrected. They show that when the classifier learned with noisy data has low confidence on the label, such a label is likely to be corrupted. Based on this, the authors here propose a label cleaning algorithm that uses ratio of posterior probabilities corresponding to the given label and network's prediction. This is further improved in [127] where the threshold is not fixed. This algorithm can handle even some types of feature-dependent label noise.

2.3 Robust Loss Functions

These approaches focus on devising novel loss functions which are inherently **robust** thereby enabling **robust risk minimization**. As discussed in Section 1.3.3, such loss functions can result in learning a classifier which would have good generalization error with respect to the clean data distribution even though the training set has label noise.

Many short-comings of risk minimization with convex loss functions under label noise are shown in [74] where the authors propose a bounded, non-convex loss function, Savage Loss, that offers robust learning. Taking cue from [67, 74] on the short-comings of using convex loss functions, the authors of [23] use a heavy tailed distribution to obtain a modified form of logistic regression. This yields a regularized risk minimization formulation with a non-convex loss function which gives the model robustness against label noise.

For a binary classification problem, a gradient-free learning-automata based stochastic optimization scheme for minimizing risk under 0-1 loss for noise-tolerant learning of hyperplane classifiers is proposed in [96]. A general notion of *robustness under label noise* and *robust loss functions* is proposed in [73]. They derive some sufficient conditions for the robustness of risk minimization under 0-1 and Mean Squared Error (MSE) losses under NULN and SLN. They also provide some counter examples to show that, in general, Exponential, Log, and Hinge loss functions are not robust against even SLN. These results are generalized in [31] to derive sufficient conditions for the family of symmetric loss functions to be robust against SLN in a binary classification problem. These theoretical results are extended to the multi-class case in [30]. Under the condition of Bayes risk being zero, they derive sufficient conditions for symmetric loss functions to be robust against general CCLN as well as some types of NULN. It is shown in [14] in case of binary classification that, for CCLN noise model as proposed in [99], symmetric margin losses are robust under minimization of Balanced Error Rate (BER) risk and Area Under Curve (AUC) risk whereas non-symmetric losses need not necessarily yield the same risk minimizers for the clean and noisy case. The authors here also propose a barrier-hinge margin loss, which is not globally symmetric but which shows robustness against label noise empirically. This work also highlights the advantages that symmetric losses offer in learning under label noise.

Based on results from [67] about the lack of robustness of convex loss functions in presence of SLN, [107] proposes a negatively-unbounded, convex, classification-calibrated loss which is a modification of the hinge loss, called unhinged loss. The authors of this paper show that the proposed unhinged loss is robust against SLN and that many convex loss functions are robust against SLN if the learner is strongly ℓ_2-regularized. A parametrized loss function, Generalized Cross-Entropy loss, is proposed in [128] and the CCE and MAE are seen as special cases of this loss. Risk minimization under this loss allows one to control the rate of convergence and degree of noise-robustness by enabling parametrized weighting for the gradient term to avoid some issues CCE and MAE losses face. MAE is robust against label noise [30] but results in a slow convergence rate owing to a constant-valued gradient. On the other hand, CCE loss gives faster convergence rates but is not robust against label noise [73]. Hence, it is proposed in [42] to use both MAE and CCE with a weighting mechanism to control the contribution of each which is done by stochastically switching between the two loss functions. It is proposed in [109] to use the reverse cross-entropy (RCE) loss function as a regularizer while using cross-entropy loss (CCE) for training, to control the underfitting to clean data. Another modification of CCE loss by using the n-th order Taylor Series expansion of CCE loss is proposed in [27] for robust learning.

2.4 Loss Correction

We explained loss Correction based methods for robust risk minimization in Section 1.3.3. Loss correction involves modifying the given loss function such that the minimizer of risk under this new, modified loss with respect to the noisy distribution would be the same as risk minimizer with the original loss function with respect to the clean distribution. However, this 'correction' to design the modification of loss function requires knowledge of the label noise rates. Hence a central issue in loss correction approaches is that of estimating the noise rates.

This idea of loss correction was first proposed for binary classification in [82] to tackle SLN. This method of loss correction was extended to the multi-class case in [86] for robustness against SLN and CCLN. They show that if one uses a special class of loss functions called proper composite losses [93], it can be guaranteed that these two minimizers are the same without requiring Equation 1.10 to hold true for all f. Estimation of the noise rate matrix is done with the help of 'anchor points'[2] which is then used for 'correction' of loss. This method is further improved in [43] by estimating noise rate matrix with an extra set of data with clean labels rather than 'anchor points' which may be noisy samples. A model-agnostic meta-learning framework is proposed in [110] to learn this noise rate matrix directly from the data rather estimating it as mentioned above. However, this method requires an extra set of clean labels for learning the matrix. For a given mini-batch, updation of noise rate matrix and loss correction are done one after the other after which network parameters are updated.

[2]An anchor point $\mathbf{x}$ for class-i is defined by $P(y^{cl}_{\mathbf{x}} = i|\mathbf{X} = \mathbf{x}) = 1$ [66, 98]. In practice one considers all instances whose computed class posterior probability is close to one as anchor points.

This estimation of noise rate matrix has been further improved in [117]. Here one starts with the noise rate matrix initially estimated using anchor points and then jointly updates the noise rate matrix and neural network parameters by minimizing a weighted loss wherein the weights depend on this modified/revised transition matrix. Another method is suggested in [36] to improve the estimation by incorporating priors on the noise rate matrix.

2.5 Regularization

This approach is one of the most popular ones and has seen a lot of interest in recent years. The idea here is to design implicit or explicit regularization methods for the training of classifier under label noise so as to reduce overfitting to the noisy data and/or underfitting to the clean data.

As mentioned in chapter 1, neural networks can learn to reproduce even random labels for the training data and this phenomenon has been called memorization. It is seen that the neural networks can learn weights to drive the training error to zero even when labels of training data are randomly altered [123]. However, they seem to be learning the clean data first before overfitting to noisy labels [6]. Based on such observations of [6, 123] about the memorization effect in neural networks, [71] proposes an algorithm that identifies the instant during the training phase whence the network has stopped learning from clean examples and has started overfitting to noisy-examples; after this point, the learning happens via bootstrapping (in a manner similar to that of [92]) wherein the network's predictions are given increasingly more weightage for supervision rather than the labels from the dataset.

A simple, data-agnostic data-augmentation method, *mixup*, is proposed in [125] which involves using convex combinations of samples and labels picked at random from training data. It encourages the classifier to behave linearly in-between the training samples thereby helping the classifier make better predictions in the vicinity of the training samples beyond the training set distribution. They empirically show that this helps with robustness against label noise. Use of this *mixup* data augmentation along with a dynamically weighted version of bootstrapping that is similar to that in [92], is proposed in [4]. Instead of using static weights for bootstrapping as in [92], the authors here use dynamically updated (after every epoch) weights obtained by learning a 2-component Beta Mixture Model (BMM). BMM is used to obtain the probability of a label being clean or not which are then used as weights for bootstrapping via EM-algorithm. A similar method with two neural networks for training is proposed in [61]. Two 2-component Gaussian Mixture Models (GMMs) for sample loss values are trained to separate data into clean and noisy data. The GMM obtained for one network is used to segregate the data for the other network. These mixture component probabilities are used as the weights for the bootstrapping loss along with data augmentation procedure inspired by [9] for training the pair of neural networks.

Comparison of feature representations of samples from noisy training set against features obtained from an auxiliary network trained with extra set of clean data is proposed in [60] to identify whether or not the training set sample has noisy labels. The cosine similarity between these features is used for regularization here. One can also use this similarity to rank samples and remove some samples based on a user-defined threshold. Apart from this, these similarities can also be used to assign sample weights to reduce overfitting to noisy data.

A regularizer that smoothens the output posterior probability distribution around training data is proposed in [79] which is shown to increase robustness against label noise. It is empirically shown in [35] that performing a gradient ascent using samples with labels likely to be noisy and gradient descent using samples believed to have clean labels aids in robustness against label noise. The heuristic for what labels are to be believed as clean or noisy has to be user-defined here. A meta-learning algorithm in the teacher-student framework is proposed in [63] wherein a consistency loss for prediction between teacher and student model predictions is used as a regularizer for achieving robustness against label noise. Instead of training or cross-training like in [37,72,122], [111] proposes a joint training of two networks using CCE loss with the contrastive loss (JS Divergence) between the two network predictions as a regularizer. It is shown in [41] that decreasing conditional mutual information of weights and labels corresponds to reducing memorization of label noise. They also propose a way to modify the gradients thereby controlling this mutual information to reduce the overfitting to noisy data. Using the distance between the network parameters from initialization as a regularizer is proposed in [45]. They also propose adding a trainable auxiliary variables to the network output for each training example as a regularizer. They theoretically prove that gradient descent training with either of these two methods leads to a generalization guarantee on the clean data distribution despite being trained using noisy labels. This analysis relies on the connection between wide neural network and neural tangent kernel (NTK). Empirical results show the effectiveness of these regularizers. Based on observations of [6] wherein neural networks are seen to be learning from clean data before overfitting to the noisy data, a regularizer is devised in [65] that maximizes the inner product between output posterior probabilities and 'target' labels. These 'target' labels are exponential average of old 'target' labels and current output posterior probabilities. It is argued in [115] that inner product between the network parameter values and corresponding gradients indicates whether or not that parameter is critical for learning from clean data. The higher the value, the more critical it is to update it so that we are less likely to learn from noisy data. A certain fraction (dependent on noise rates) of parameters is marked as non-critical which are updated only using weight decay that will help us drive them to zero value thereby helping improve generalization.

An empirical study of a family of f-divergences for providing robustness against label noise is presented in [112]. Adding zero-mean Gaussian noise to labels (stochastic label noise) for SGD is shown to be effective for robustness to label noise in [15].

2.6 Sample Reweighting

This approach can also be viewed as part of the the broad *Regularization* category that we have talked about in the previous section. Since this thesis proposes a new algorithm for sample reweighting, we review the literature for this category separately here. We also discuss some of these issues in the next chapter.

The basic idea in sample reweighting is about minimizing a weighted loss wherein weight for each sample is adjusted such that more weightage is given to data that is believed to be 'clean'. This is done to reduce overfitting to noisy data and increase the influence of clean data. The notion of 'clean' or 'noisy' has to be defined by the algorithm.

Under the assumption that the classifier predictions can be trusted more as epochs go by, an algorithm is proposed in [92] that uses a convex combination of noisy training labels and current prediciton probabilities of the classifier as the 'soft' labels for training with CCE loss. This yields a heuristic notion of 'consistency' evaluation of noisy samples. This formulation is equivalent to softmax regression with minimum entropy regularization as studied in [32] which encourages the model to have a high confidence in predicted labels. A similar method is proposed in [38] with the difference being that here the convex combination of noisy and 'corrected' labels is used. The 'corrected' labels here are obtained by identifying multiple prototypes for each class that are more likely to have clean labels which are then used to find the most-likely correct label for a given sample.

A simple change in neural network parameter updation rule is proposed in [72]. In standard training with clean data, the network will make fewer mistakes and hence improves its predictions as epochs pass by. But, in case of training with noisy data, since the parameters will be updated based on mistakes in predicting noisy labels, authors propose using two similar networks and perform parameter updates only with samples upon which their predictions disagree. This preserves the behaviour of standard learning wherein the parameters are updated more frequently at the beginning and less so towards the end of the training. Moreover, mismatch in prediction and noisy labels is not used for selecting samples for parameter updation which helps reduce overfitting to the noisy labels. An algorithm is proposed in [19] which uses an auxiliary network learned from extra data with clean label whose output is to be used as sample weights for the classifier learning under label noise. This is similar to [50] with the difference being that here the updates are done in a meta-learning fashion and the sample weights are not binary. It is suggested in [77, 89] that preferring samples with low loss values helps in learning in presence of noisy data whereas it is observed in [1, 29] that preferring high loss-valued samples helps accelerate and stabilize training with Stochastic Gradient Descent (SGD). Based on these observations and the fact that it's often not known how noisy the data is, an algorithm is proposed in [13] that selects samples either on the basis of variance in posterior probabilities for the class or proximity of true class posterior probabilities to the decision threshold. Another method proposed in [50] pre-trains an auxiliary network using extra data with clean labels which serves as a sample selection function for training the main network. Inspired by Co-Training [11], a sample selection scheme is proposed in [37] wherein two networks select a fraction of small-loss valued samples (dependent on label noise rates) and feed the selected samples to other network for training. An improved version of this method is proposed in [122] by selecting samples in the same manner but from a subset of samples upon which the two networks' predictions disagree. A similar algorithm is proposed in [100] wherein only a fraction of small-loss valued samples are selected for training. Another algorithm, proposed in [120], improves Co-Teaching [37] by solving a bilevel optimization problem with the help of AutoML techniques to obtain a relatively optimal sample fraction value. It is shown in [16] that randomly dividing noisy training set and using cross-validation to identify clean samples offers robustness. Under the classical assumption that data from the same class should form a cluster and be similar to each other[3], it is proposed in [22] to use a distance measure of sample from a set of cluster centers for each class in the training data to decide on the sample weights. The samples which are farther from the set of cluster centers for corresponding class will be deemed as noisy samples here and therefore will have a smaller sample weight.

Meta-learning based framework to compute and update sample weights and neural network parameters is proposed in [94]. The weights are computed with the help of inner product between gradients on training samples and gradients on validation samples which have clean labels. The algorithms proposed in [24, 47, 64] solve a bi-level optimization problem for learning sample weights and network parameters that results in a similar inner product between gradients of samples for robustness against label noise. However, one doesn't require an extra clean validation set here. The method proposed in [101] improves upon that in [94] by learning a sample reweighting function that maps loss values to sample weights via an auxiliary network. This is similar to the idea of MentorNet [50] with the difference being that MentorNet is not based on meta-learning. An importance sampling based sample reweighting approach is proposed in [66,97]; however one needs to estimate these noise rates with the help of extra validation set. The

[3]Clustering hypothesis: https://en.wikipedia.org/wiki/Cluster_hypothesis

method in [62] devises a weighting scheme taking into account class-imbalance, intra-class diversity and inter-class similarity. A sample selection method is proposed in [78] that identifies subsets of training set that has an approximately low-rank Jacobian. They show that this helps increase robustness against label noise. Based on the intuition that noisy data is isolated and surrounded by clean data and clean data well-separated into clusters, [114] proposes a sample selection algorithm that leverages this topographical structure of training data.

2.7 Conclusions

In this chapter we have provided a reasonably extensive overview of the literature on learning under label noise. We categorized the available methods into six categories and under each approach tried to give an idea of the different methods proposed. As is clear from all the preceding sections, there are a large number of different methods suggested for this problem. This area has been particularly active in the last five years. Many proposed methods are essentially empirical in the sense that there are very few methods that give theoretical guarantees on robustness. In the next chapter we present a new sample selection algorithm.

Chapter 3

Adaptive Sample Selection

As discussed in the previous chapter, there have been many different approaches that are explored for robust learning of classifiers under label noise. In this chapter we focus on one of these approaches, namely, sample reweighting. We present a new sample selection heuristic that results in a simple and efficient algorithm whose robustness properties are comparable to or better than the state-of-art.

3.1 Introduction

In the last few years, many algorithms based on sample reweighting are proposed for robust learning. The sample reweighting approach consists of assigning weights to different samples and then minimizing the weighted loss. The weights can be binary or real-valued. When they are binary, we call it a sample selection algorithm. Here, the motivation for the algorithm is to guess which samples have incorrect labels, and then consider the loss only on the remaining samples while learning the parameters of the neural network. Even in the case where the weights are real valued, the motivation is the same: reduce the effect of samples with incorrect labels on the learning of the neural network. There have been many heuristics suggested for choosing such weights (e.g., [37, 122]). There are also algorithms that use an auxiliary network to learn the appropriate weights using a separate set of samples with clean labels (e.g., [50, 101]). Another possible approach is to optimize both on the weights for samples as well as on the parameters of the neural network (e.g., [94, 104]).

In this chapter we present a novel algorithm that adaptively selects samples based on the statistics of observed loss values (or posterior probability values) in a minibatch. The algorithm is very simple and achieves good robustness to label noise. The algorithm does not use any additional system for learning weights for samples, does not need extra data with clean labels and does not assume any knowledge of noise rates. Since it is a sample selection algorithm, it essentially amounts to choosing those samples in a mini-batch whose current loss values are below some threshold. The threshold itself is determined based on the statistics of current mini-batch and is thus automatically adapted as the learning progresses. Like many of the sample reweighting algorithms, the proposed algorithm also can be seen as motivated by curriculum learning.

The curriculum learning [8, 58] is a general strategy of sequencing of samples so that the networks learn the 'easy' samples well before learning the 'hard' ones. The curriculum or sequencing of the samples can be done through an external teacher or through a mechanism in the algorithm itself (e.g., self-paced learning [58]). This is often brought about by giving different weights to different samples in the training set. Many of the recent algorithms for robust learning based on sample reweighting can be seen as motivated by a similar idea. In the context of label noise, one can think of clean samples as the easy ones and the samples with wrong labels as the hard ones. In many of these approaches, the weight assigned to a sample is essentially determined by the loss function value on that sample with a heuristic that, low loss values indicate reliable labels. Many different ways of fixing/learning such weights have been proposed (e.g., [47,94,101]). A good justification for this approach of assigning weights to samples for achieving robustness comes from some recent studies on the effects of noisily-labelled data on learning deep neural networks. It is empirically shown in [6] that deep neural networks can learn to achieve zero training error on completely randomly labelled data, a phenomenon termed as 'memorization'. However, further studies such as [6, 71] have shown that the networks, when trained on randomly-labelled data, seem to learn from the cleanly-labelled data first before overfitting to the noisily-labelled data. Thus, till the overfitting to noisy labels starts, the loss of samples with noisy labels are likely to be higher than that for samples with clean labels.

Motivated by this, several strategies of 'curriculum learning' have been devised that aim to select (or give more weightage to) ‘clean’ samples for obtaining a degree of robustness against label noise [37,50,68,120,122]. All such methods essentially employ the heuristic of ‘small loss’ for sample selection or weighting wherein (a fraction of) small-loss valued samples are preferentially used for learning the network.

Algorithms such as Co-Teaching [37] and Co-Teaching+ [122] use two networks and select samples with loss value below a threshold in one network to train the other. In Co-Teaching, the threshold is computed based on the knowledge of noise rate and the computed threshold is the same for all samples irrespective of the class labels. The same threshold computation method is used in Co-Teaching+ but the sample selection is based on disagreement between the two networks. That is, among the low-loss valued samples, the subset on which the two networks differ are the ones selected to train the networks.

The MentorNet [50], another recent algorithm based on curriculum learning, uses an auxiliary neural network trained to serve as a sample selection function. One needs a separate set of samples with clean labels to train this second network, termed the mentor network.

There are other adaptive sample reweighting schemes such as [94,101] which also rely on learning sample weights. They solve a separate learning problem where, given the current loss values one needs to infer appropriate weights for all samples. These methods also require some extra data with clean labels for learning these weights.

These algorithms which employ an additional learning step to infer a mapping from loss values to weights of samples, need additional computing resources as well as access to extra data with clean labels. In addition, they have extra hyperparameters for training the sample weighting function and these also need tuning. All such methods, in effect, assume that one can assess whether or not a sample has clean label based on some function of the loss value of that sample. However, loss value of any specific sample is itself a function of the current state of learning and it evolves with epochs. Loss values of even clean samples may change over a significant range during the course of learning. Further, the loss values achievable by a network even on clean samples may be different for samples of different classes.

The algorithms such as Co-Teaching and Co-Teaching+ also have threshold computing method which does not distinguish between samples of different classes. The calculation of the loss threshold needs knowledge of noise rate. These algorithms use a single number as the noise rate. That is reasonable for symmetric label noise. However, when we have class conditional label noise, it is not clear what noise rate one should use in these algorithms. These methods rely on estimating the noise rate from samples. In general, it is difficult to estimate label corruption probabilities reliably from the training data. As we show through simulations here, the method is sensitive to errors in the estimation of noise rate.

Motivated by these considerations, we propose a simple, adaptive curriculum based sample selection strategy called *BAtch REweighting* (**BARE**). The idea is to focus on the current state of learning, in a given mini-batch, for identifying the noisily labelled data in it. The statistics of loss values of all samples in a mini-batch would give useful information on current state of learning. Our algorithm utilizes these batch statistics to compute the threshold for sample selection in a given mini-batch. We do not need any knowledge of noise rates at all. This will give us what is essentially a dynamic or adaptive curriculum. In addition, while calculating the batch statistics we take into consideration the class labels also and hence the dynamic thresholds are also dependent on the given labels of the samples.

The main contribution of this chapter is an adaptive sample selection strategy for robust learning that is simple and efficient, does not need any clean validation data, needs no knowledge at all of the noise rates and also does not have any hyperparameters in the sample selection process. We empirically demonstrate the effectiveness of our algorithm on benchmark datasets: MNIST [59], CIFAR-10 [56], and Clothing-1M [118]; and show that our algorithm is much more efficient in terms of time and has as good or better robustness compared to other algorithms for different types of label noise and noise rates.

3.1.1 Related Work

Curriculum learning (CL) as proposed in [8] is the designing of an optimal sequence of training samples to improve the model's performance. The order of samples in this sequence is to be decided based on a notion of *easiness* which can be fixed based on some prior knowledge. A curriculum called Self-Paced Learning (SPL) is proposed in [58] wherein easiness is decided based on how small the loss values are. A framework to unify CL and SPL is proposed in [49] by incorporating the prior knowledge about curriculum and feedback from the model during training with the help of *self-paced functions* that are to be used as regularizers. SPL with diversity [48] improved upon SPL by proposing a sample selection scheme by encouraging selection of a diverse set of *easy* samples for learning with the help of a group sparsity regularizer. This is further improved in [133] by encouraging more exploration during early phases of learning.

Motivated by similar ideas, many sample reweighting algorithms are proposed for tackling label noise in neural networks. Sample selection / reweighting algorithms for robust deep learning can be viewed as designing a fixed or adaptive curriculum. A sample selection algorithm based on the 'small loss' heuristic wherein the algorithm will select a fraction of small loss valued samples for training is proposed in [37, 122]. Two networks are cross-trained with samples selected by each other based on this criterion. [68] also relies on 'small loss' heuristic but the threshold for sample selection is adapted based on the knowledge of label noise rates. When a (small amount of) separate data with clean labels is available, [50] proposes a data-dependent, adaptive curriculum learning method wherein an auxiliary network trained on this clean data set is used to select reliable samples from the noisy training data for training the classifier. When such clean data is not available, it reduces to a non-adaptive, self-paced learning scheme. Another sample selection algorithm is proposed in [72] where the idea is to train two networks and update the network parameters only in case of a disagreement between the two networks. [68] propose a curriculum that's adaptive in spite of the threshold being fixed. This threshold, however, is dependent on the noise rate. For this, the authors propose a loss function that can be constructed using a surrogate loss of 0-1 loss function. This resultant loss is a (binary) weighted sum of the chosen surrogate loss values such that it's minimized with respect to the user-defined threshold. These sample selection functions are mostly hand-crafted and, hence, they can be sub-optimal. A general strategy is to solve a bilevel optimization problem to find the optimal sample weights. For instance, the sample selection function used in [37, 122] is sub-optimally chosen for which [120] proposes an AutoML-based approach to find a better function, by fine-tuning on separate data with clean labels. Sample reweighting algorithms proposed in [94] and [101] use online meta-learning and need some extra data with clean labels. The method in [94] uses the gradients of loss on noisy and clean data to learn the weights for samples while the method in [101] tries to learn these weights as a function of their loss values.

3.2 Batch Reweighting Algorithm

In this section we describe the proposed sample reweighting algorithm that relies on mini-batch statistics. First we recall our notation.

The distribution of data under clean labels is $\mathcal{D}$ on $\mathcal{X} \times \mathcal{Y}$. The ideal (and unobservable) training set with clean labels is $S^{cl} = \{(\mathbf{x}_i, y_i^{cl}),\ i = 1, 2, \cdots, m\}$ which consists of iid samples drawn according to $\mathcal{D}$. We consider a K-class problem and take y_i^{cl} to be one-hot vectors. We denote by e_k the one-hot vector corresponding to class k. The training data with noisy labels that we have access to is $S = \{(\mathbf{x}_i, y_i),\ i = 1, 2, \cdots, m\}$ drawn according to a distribution $\mathcal{D}_\eta$. Here we will consider only symmetric and class-conditional label noise. Hence, the y_i and y_i^{cl} are related as

$$P[y_i = e_{k'} \mid y_i^{cl} = e_k] = \eta_{kk'}$$

For the special case of symmetric noise we have $\eta_{kk} = (1 - \eta)$ and $\eta_{kk'} = \frac{\eta}{K-1}, \forall k' \neq k$. Here, we assume $\eta < \frac{K-1}{K}$. For the general class conditional noise, we assume that the noise rate matrix, N, is diagonally dominant; that is, $\eta_{kk} > \eta_{kk'}, \forall k' \neq k$.

For assessing robustness, we test the classifier on test data with clean labels. That is, test data is drawn from $\mathcal{D}$.

We denote by $f(\cdot;\theta)$ a classifier function parameterized by θ. We assume that the neural network classifiers that we use have softmax output layer. Hence, while the training set labels, y_i, are one-hot vectors, we will have $f(\mathbf{x};\theta) \in \Delta^{K-1}$, where $\Delta^{K-1} \subset [0,1]^K$ is the probability simplex. We denote by $\mathcal{L}(f(\mathbf{x};\theta), y)$ the loss function used for the classifier training which in our case is the CCE loss.

3.2.1 Adaptive Curriculum through Batch Statistics

General curriculum learning can be viewed as minimization of a weighted loss [50, 58]:

$$\begin{aligned}\min_{\theta, \mathbf{w} \in [0,1]^m} \mathcal{L}_{\text{wtd}}(\theta, \mathbf{w}) &= \sum_{i=1}^{m} w_i \mathcal{L}(f(\mathbf{x}_i; \theta), y_i) \\ &\quad + G(\mathbf{w}) + \beta ||\theta||^2\end{aligned}$$

where $G(\mathbf{w})$ represents the curriculum. Since one normally employs SGD for learning, we will take m here to be the size of a mini-batch. One simple choice for the curriculum is [58] $G(\mathbf{w}) = -\lambda ||\mathbf{w}||_1,\ \lambda > 0$. Putting this in the above, omitting the regularization term and taking $l_i = \mathcal{L}(f(\mathbf{x}_i; \theta), y_i)$, the optimization problem becomes

$$\begin{aligned}\min_{\theta, \mathbf{w} \in [0,1]^m} \mathcal{L}_{\text{wtd}}(\theta, \mathbf{w}) &= \sum_{i=1}^{m} (w_i l_i - \lambda w_i) \\ &= \sum_{i=1}^{m} (w_i l_i + (1 - w_i)\lambda) - m\lambda\end{aligned}$$

Under the usual assumption that loss function is non-negative, for the above problem, the optimal $\mathbf{w}$ for any fixed θ is: $w_i = 1$ if $l_i < \lambda$ and $w_i = 0$ otherwise. If we want an adaptive curriculum, we want λ to be dynamically adjusted based on the current state of learning. First, let us consider the case where we make λ depend on the class label. The optimization problem becomes

$$\begin{aligned}\min_{\theta, \mathbf{w} \in [0,1]^m} \mathcal{L}_{\text{wtd}}(\theta, \mathbf{w}) &= \sum_{i=1}^{m} (w_i l_i - \lambda(y_i) w_i) \\ &= \sum_{j=1}^{K} \sum_{i: y_i = e_j} (w_i l_i - \lambda_j w_i) \\ &= \sum_{j=1}^{K} \sum_{i: y_i = e_j} (w_i l_i + (1 - w_i)\lambda_j) - \sum_{j=1}^{K} \sum_{i: y_i = e_j} \lambda_j\end{aligned}$$

where $\lambda_j = \lambda(e_j)$. As is easy to see, the optimal w_i (for any fixed θ) are still given by the same relation: for an i with $y_i = e_j$, $w_i = 1$ when $l_i < \lambda_j$. Note that this relation for optimal w_i is true even if we make λ_j a function of θ and of all $\mathbf{x}_i$ with $y_i = e_j$. Thus we can have a truly dynamically adaptive curriculum by making these λ_j depend on all $\mathbf{x}_i$

of that class in the mini-batch and the current θ.

The next question is how we should decide or evolve these λ_j. As we mentioned earlier, we want these to be determined by the statistics of loss values in the mini-batch.

Consider those i for which $y_i = e_j$. We would be setting $w_i = 1$ and hence use this sample to update θ in this minibatch if this $l_i < \lambda_j$. We want λ_i to be fixed based on the observed loss values of this mini-batch. Since there is sufficient empirical evidence that we tend to learn from the clean samples before overfitting to the noisy ones, some quantile or similar statistic of the set of observed loss values in the mini-batch (among patterns labelled with a specific class) would be a good choice for λ_j.

Since we are using CCE loss, we have $l_i = -\ln(f_j(\mathbf{x}_i;\theta))$ and as the network has softmax output layer, $f_j(\mathbf{x}_i;\theta)$ is the posterior probability of class-j under current θ for $\mathbf{x}_i$. Since the loss and this posterior probability are inversely related, our criterion for selection of a sample could be that the assigned posterior probability is above a threshold which is some statistic of the observed posterior probabilities in the mini-batch. We take the statistic to be mean plus one standard deviation. (This choice does not appear to be too critical and we got comparable results with median as the statistic)

We can sum up the above discussion of our method of adaptive curriculum based on mini-batch statistics as follows. In any mini-batch we set the weights for samples as

$$w_i = \begin{cases} 1 & \text{if } f_{y_i}(\mathbf{x}_i;\theta) \geq \frac{1}{|\mathcal{S}_{y_i}|}\sum_{s\in\mathcal{S}_{y_i}} f_{y_s}(\mathbf{x}_s;\theta) + \sigma_{y_i} \\ 0 & \text{else} \end{cases} \tag{3.1}$$

where $\mathcal{S}_{y_i} = \{k \in [m] \mid y_k = y_i\}$ and σ_{e_j} indicates the sample variance of the class posterior probabilities for class-j in the given mini-batch.

Algorithm Implementation

Keeping in mind that the neural networks are trained in a mini-batch manner, Algorithm 1 consists of three parts: i.) computing sample selection threshold, $\Lambda_{K\times 1}$, for a given mini-batch of data (Step 8), ii.) sample selection based on this threshold (Steps 9-13) as per Equation 3.1, and iii.) parameter updation using these selected samples (Step 14).

Algorithm 1: BAtch REweighting (BARE) Algorithm

```
1: Input: noisy dataset $\mathcal{D}_\eta$, # of classes $K$, # of epochs $T_{max}$, learning rate $\alpha$
2: Initialize: Network parameters, $\theta_0$
3: for $t = 0$ to $T_{max} - 1$ do
4:     Shuffle the training dataset $\mathcal{D}_\eta$
5:     for $i = 1$ to $|\mathcal{D}_\eta|/|\mathcal{M}|$ do
6:         Draw a mini-batch $\mathcal{M}$ from $\mathcal{D}_\eta$
7:         $\mathcal{R} \leftarrow \phi$    /* selected samples in $\mathcal{M}$ */
8:         For $\mathcal{M}$, compute vector, $\Lambda_{K\times 1}$ where $\Lambda_p = \frac{1}{|\mathcal{S}_p|}\sum_{s\in\mathcal{S}_p} f_p(\mathbf{x}_s;\theta_t) + \sigma_p$ /*(Eq.
           3.1), average posterior probability */
9:         for each $\mathbf{x} \in \mathcal{M}$ do
10:            if $f_{y_\mathbf{x}}(\mathbf{x};\theta_t) \geq \Lambda_{y_\mathbf{x}}$ then
11:                $\mathcal{R} \leftarrow \mathcal{R} \cup (\mathbf{x}, y_\mathbf{x})$    /* Select sample */
12:            end if
13:        end for
14:        $\theta_{t+1} = \theta_t - \alpha\nabla\left(\frac{1}{|\mathcal{R}|}\sum_{(\mathbf{x},y_\mathbf{x})\in\mathcal{R}} \mathcal{L}(\mathbf{x}, y_\mathbf{x};\theta_t)\right)$    /* parameter updates */
15:    end for
16: end for
17: Output: $\theta_t$
```

3.3 Experiments on Noisy Dataset

Dataset: We demonstrate the effectiveness of the proposed algorithm on two benchmark image datasets: MNIST and CIFAR10. These data sets are used to benchmark almost all algorithms for robust learning under label noise and we briefly describe the data sets. (The details of data sets are given in Table 3.1). MNIST contains 60,000 training images and 10,000 test images (of size 28×28) with 10 classes. CIFAR-10 contains 50,000 training images and 10,000 test images (of size 32×32) with 10 classes.

We test the algorithms on two types of label noise: symmetric and class-conditional label noise. In symmetric label noise, each label is randomly flipped to any of the remaining classes with equal probability, whereas for class-conditional noise, label flipping is done in a set of similar classes. For the simulations here, for MNIST, the following flipping is done: $1 \leftarrow 7$, $2 \rightarrow 7$, $3 \rightarrow 8$, and $5 \leftrightarrow 6$. Similarly, for CIFAR10, the following flipping is done: TRUCK $\rightarrow$ AUTOMOBILE, BIRD $\rightarrow$ AIRPLANE, DEER $\rightarrow$ HORSE, CAT $\leftrightarrow$ DOG. We use this type of noise because that is arguably a more realistic scenario and also because it is the type of noise, in addition to symmetric noise, that other algorithms for learning under label noise have used. Apart from this, we also provide results with an arbitrary noise rate matrix. For all the datasets, 80% of the training set is used for training and, from the remaining 20% data, we sample 1000 images that constitute the validation set.

We also experiment with the Clothing-1M dataset [118] which is a large-scale dataset obtained by scraping off the web for different images related to clothing. It contains noise that can be characterized as somewhat close to feature-dependent noise, the most generic kind of label noise. An estimated 40% images have noisy labels. The training dataset contains 1 million images and the number of classes is 14. There are additional training, validation, and test sets of 50k, 14k, and 10k images respectively with clean labels. Since there's a class imbalance, following similar procedure as in existing baselines, we use 260k images from the original noisy training set for training while ensuring equal number of images per class in the set. We use a test set of 10k images with clean labels for performance evaluation.

Data Augmentations: For MNIST we use no data augmentation. For CIFAR-10 we do a random cropping with padding of 4, and random horizontal flips. For Clothing-1M we do random cropping while ensuring image size is fixed.

Table 3.1: Dataset details

	TRAIN SIZE	TEST SIZE	# CLASS	SIZE
MNIST	60,000	10,000	10	28×28
CIFAR-10	50,000	10,000	10	32×32
CLOTHING-1M	10,00,000	10,000	14	224×224

Baselines: We compare the proposed algorithm with the following algorithms from literature:

(1.) Co-Teaching (CoT) [37] which involves cross-training of two similar networks by selecting a fraction (dependent on noise rates) of low loss valued samples;

(2.) Co-Teaching+ (CoT+) [122] which improves upon CoT with the difference being sample selection only from the subset upon which the two networks' predictions disagree;

(3.) Meta-Ren (MR) [94], which involves meta-learning of sample weights on-the-fly by comparing gradients for clean and noisy data;

(4.) Meta-Net (MN) [101], which improves upon MR by explicitly learning sample weights via a separate neural network;

(5.) Curriculum Loss (CL) [68], which involves a curriculum for sample selection based on (estimated) noise rates;

(6.) Standard (CCE), which is the usual training through empirical risk minimization with cross-entropy loss (using the data with noisy labels).

Among these baselines, CoT, CoT+, and CL are sample selection algorithms that require knowledge of noise rates. The algorithms CoT+ and CL need a few initial iterations without any sample selection as a warm-up period; we used 5 epochs and 10 epochs as warm up period during training for MNIST and CIFAR-10 respectively. MR and MN assume access to a small set of clean validation data. Because of this, and for a fair comparison among all the baselines, a clean validation set of 1000 samples is used in case of MR and MN, and the same set of samples but with the noisy labels is used for the rest of the algorithms including the proposed one.

Network architectures & Optimizers: While most algorithms for learning under label noise use MNIST and CIFAR10 data, different algorithms use different network architectures. Here we have decided to use small networks that give state of art performance on clean data and investigate the robustness we get by using our algorithm. We use one MLP and one CNN architecture. For MNIST we train a 1-hidden layer fully-connected network with Adam (learning rate $= 2 \times 10^{-4}$ and a learning rate scheduler: ReduceLROnPlateau). (This is same as the network used for Co-Teaching and Co-Teaching+ [37, 122]). For CIFAR-10 we train a 4-layer CNN with Adam [55] (learning rate = 2×10^{-3} and a learning rate scheduler: ReduceLROnPlateau). All networks are trained for 200 epochs. For MR, SGD optimizer with momentum 0.9 and learning rate of 1×10^{-3} is used as the meta-optimizer. For MN, SGD optimizer with learning rate of 2×10^{-3} is used as meta-optimizer. For CL, soft hinge loss is used as suggested in [68] instead of cross-entropy loss. Rest of the algorithms implemented in this chapter use cross-entropy loss. All the simulations are run for 5 trials. A pre-trained ResNet-50 is used for training on Clothing-1M with SGD (learning rate of 1×10^{-3} that is halved at epochs 6 and 11) with a weight decay of 1×10^{-3} and momentum 0.9 for 14 epochs. All experiments use PyTorch [85], NumPy [40], scikit-learn [88], and NVIDIA Titan X Pascal GPU with CUDA 10.0. Table 3.2 contains details about the network architecture used for training on MNIST and CIFAR-10 datasets. These settings of optimizer, learning rate, and learning rate scheduler were found to work the best for our experimental and hardware setup. All the codes for these experiments are available here: https://github.com/dbp1994/masters_thesis_codes/tree/main/BARE.

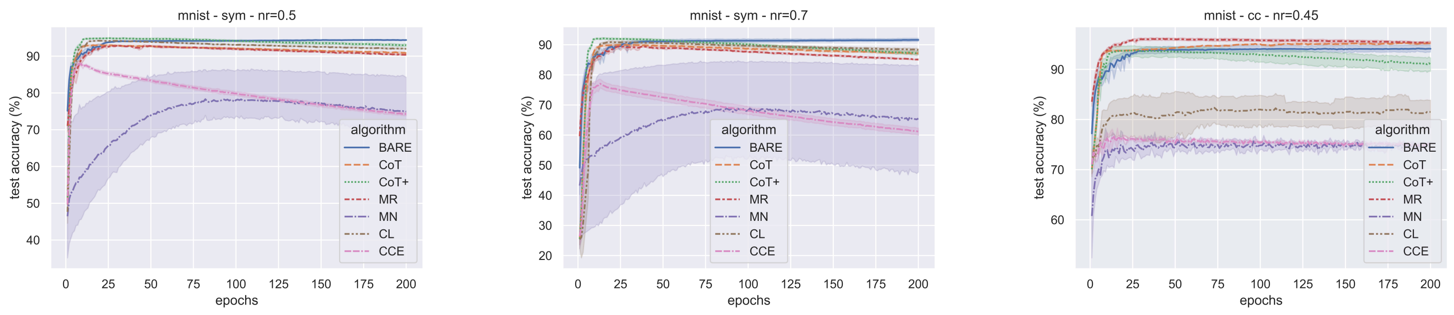
Table 3.2: Network Architectures used for training on MNIST and CIFAR-10 datasets

MNIST	CIFAR-10
	3×3 conv., 64 ReLU, stride 1, padding 1
	Batch Normalization
	2×2 Max Pooling, stride 2
	3×3 conv., 128 ReLU, stride 1, padding 1
	Batch Normalization
dense 28×28 → 256	2×2 Max Pooling, stride 2
	3×3 conv., 196 ReLU, stride 1, padding 1
	Batch Normalization
	3×3 conv., 16 ReLU, stride 1, padding 1
	Batch Normalization
	2×2 Max Pooling, stride 2
dense 256 → 10	dense 256 →10

Performance Metrics: For all algorithms we compare test accuracies on a separate test set with clean labels. The main idea in all sample selection schemes is to identify noisy labels. Hence, in addition to test accuracies, we also compare precision (# *clean labels selected / # of selected labels*) and recall (# *clean labels selected / # of clean labels in the data*) in identifying noisy labels.

3.3.1 Discussion of Results

Performance on MNIST



(a) (b) (c)

Figure 3.1: Test Accuracies - MNIST - Symmetric ((a) & (b)) & Class-conditional ((c)) Label Noise

Figure 3.1 shows the evolution of test accuracy (with training epochs) under symmetric ($\eta \in \{0.5, 0.7\}$) and class conditional ($\eta = 0.45$) label noise for different algorithms. We can see from the figure that the proposed algorithm outperforms the baselines for symmetric noise. For the case of class-conditional noise, the test accuracy of the proposed algorithm is marginally less than the best of the baselines, namely CoT and MR.

Figure 3.1 showed the evolution of test accuracies with epochs. We tabulate the final test accuracies of all algorithms in Tables 3.3 – 3.5. The best two results are in bold. These are accuracies achieved at the end of training. For CoT [37] and CoT+ [122], we show accuracies only of that network which performs the best out of the two that are trained.

Table 3.3: Test Accuracy (%) for MNIST - $\eta = 0.5$ (symmetric)

ALGORITHM	TEST ACCURACY
CoT [37]	90.80 ± 0.18
CoT+ [122]	**93.17 ± 0.3**
MR [94]	90.39 ± 0.07
MN [101]	74.94 ± 9.56
CL [68]	92.00 ± 0.26
CCE	74.30 ± 0.55
BARE (Ours)	**94.38 ± 0.13**

Table 3.4: Test Accuracy (%) for MNIST - $\eta = 0.7$ (symmetric)

ALGORITHM	TEST ACCURACY
CoT [37]	87.17 ± 0.45
CoT+ [122]	87.26 ± 0.67
MR [94]	85.10 ± 0.28
MN [101]	65.52 ± 21.35
CL [68]	**88.28 ± 0.45**
CCE	61.19 ± 1.29
BARE (Ours)	**91.61 ± 0.60**

Performance on CIFAR-10

Figure 3.2 shows the test accuracies of the various algorithms as the training progresses for both symmetric ($\eta \in \{0.3, 0.7\}$) and class-conditional ($\eta = 0.4$) label noise. We can see from the figure that the proposed algorithm outperforms the baseline schemes and its test accuracies are uniformly good for all types of label noise.

It is to be noted that while test accuracies for our algorithm stay saturated after attaining maximum performance, the other algorithms' performance seems to deteriorate as can be seen in the form of accuracy dips towards the end of training. This suggests

Table 3.5: Test Accuracy (%) for MNIST - $\eta = 0.45$ (class-conditional)

ALGORITHM	TEST ACCURACY
CoT [37]	$\mathbf{95.20 \pm 0.22}$
CoT+ [122]	91.10 ± 1.51
MR [94]	$\mathbf{95.40 \pm 0.31}$
MN [101]	75.03 ± 0.59
CL [68]	81.52 ± 3.27
CCE	74.96 ± 0.21
BARE (Ours)	94.11 ± 0.77

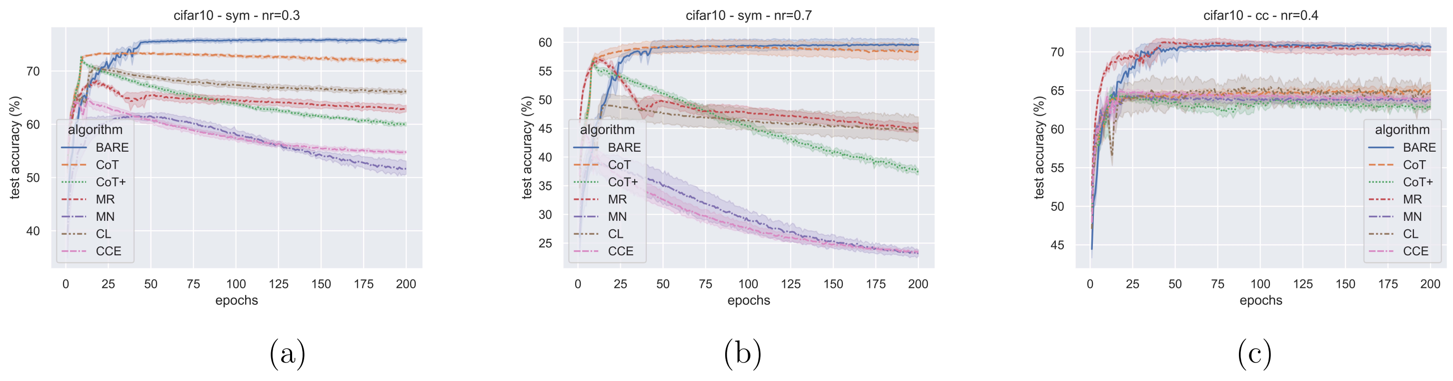


Figure 3.2: Test Accuracies - CIFAR10 - Symmetric ((a) & (b)) & Class-conditional ((c)) Label Noise

that our proposed algorithm doesn't let the network overfit even after long durations of training unlike the case with other algorithms.

All the algorithms, except the proposed one, have hyperparameters (in the sample selection/weighting method) and the accuracies reported here are for the best possible hyperparameter values obtained through tuning. The MR and MN algorithms are particularly sensitive to hyperparameter values in the meta learning algorithm. In contrast, BARE has no hyperparameters for the sample selection and hence no such tuning is involved.

As in the case of MNIST, we tabulate the final test accuracies of all algorithms on

CIFAR in Tables 3.6 – 3.8. The best two results are in bold.

Table 3.6: Test Accuracy (%) for CIFAR-10 - $\eta = 0.3$ (symmetric)

Algorithm	Test Accuracy
CoT [37]	$\mathbf{71.72 \pm 0.30}$
CoT+ [122]	60.14 ± 0.35
MR [94]	62.96 ± 0.70
MN [101]	51.65 ± 1.49
CL [68]	66.124 ± 0.45
CCE	54.83 ± 0.28
BARE (Ours)	$\mathbf{75.85 \pm 0.41}$

Table 3.7: Test Accuracy (%) for CIFAR-10 - $\eta = 0.7$ (symmetric)

Algorithm	Test Accuracy
CoT [37]	$\mathbf{58.95 \pm 1.31}$
CoT+ [122]	37.69 ± 0.70
MR [94]	45.14 ± 1.04
MN [101]	23.23 ± 0.65
CL [68]	44.82 ± 2.42
CCE	23.46 ± 0.37
BARE (Ours)	$\mathbf{59.53 \pm 1.12}$

It may be noted from the tables giving test accuracies on MNIST and CIFAR-10, that sometimes the standard deviation in the accuracy for MN is high. As we mentioned earlier, we noticed that MN is very sensitive to the tuning of hyper parameters. While we tried our best to tune all the hyper parameters, may be the final ones we found for these cases are still not the best and that is why the standard deviation is high.

Table 3.8: Test Accuracy (%) for CIFAR-10 - $\eta = 0.4$ (class-conditional)

Algorithm	Test Accuracy
CoT [37]	65.26 ± 0.78
CoT+ [122]	63.05 ± 0.39
MR [94]	$\mathbf{70.27 \pm 0.77}$
MN [101]	63.84 ± 0.41
CL [68]	64.48 ± 2.02
CCE	64.06 ± 0.32
BARE (Ours)	$\mathbf{70.63 \pm 0.46}$

Table 3.9: Algorithm run times for training (in seconds)

Algorithm	MNIST	CIFAR10
BARE	**310.64**	**930.78**
CoT	504.5	1687.9
CoT+	537.7	1790.57
MR	807.4	8130.87
MN	1138.4	8891.6
CL	730.15	1254.3
CCE	**229.27**	**825.68**

Efficiency of BARE

Table 3.9 shows the typical run times for 200 epochs of training with all the algorithms. It can be seen from the table that the proposed algorithm takes roughly the same time as the usual training with CCE loss whereas all other baselines are significantly more expensive computationally. In case of MR and MN, the run times are around 8 times that of the proposed algorithm for CIFAR-10.

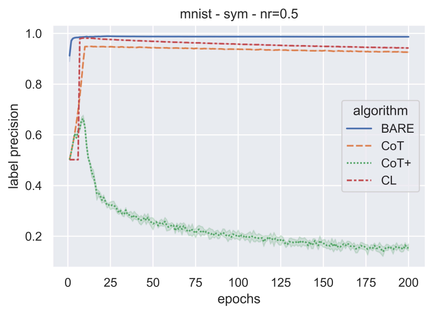
Performance on Clothing1M

The results are summarized in Table 3.10. We report the baseline accuracies as observed in the corresponding papers. And, for this reason, the baselines with which we compare our results are different. These are the baselines that have reported results on this dataset. It shows that that even for datasets used in practice which have label noise that isn't synthetic unlike the symmetric and class-conditional label noise used for aforementioned simulations, the proposed algorithm performs better than all but one baselines. However, it is to be noted that DivideMix requires about 2.4 times the computation time required for BARE. In addition to this, DivideMix requires tuning of 5 hyperparameters whereas no such tuning is required for BARE.

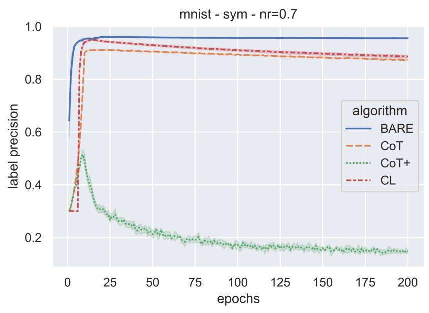
Table 3.10: Test accuracies on Clothing-1M dataset

Algorithm	Test Accuracy (%)
CCE	68.94
D2L [71]	69.47
GCE [128]	69.75
Forward [86]	69.84
CoT [37][1]	70.15
JoCoR [111]	70.30
SEAL [17]	70.63
DY [5]	71.00
SCE [109]	71.02
LRT [132]	71.74
PTD-R-V [116]	71.67
Joint Opt. [104]	72.23
BARE (Ours)	**72.28**
DivideMix [61]	74.76

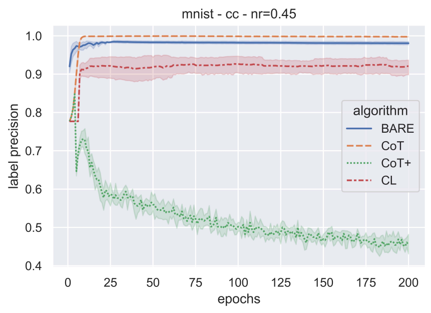
[1]as reported in [17]



(a)

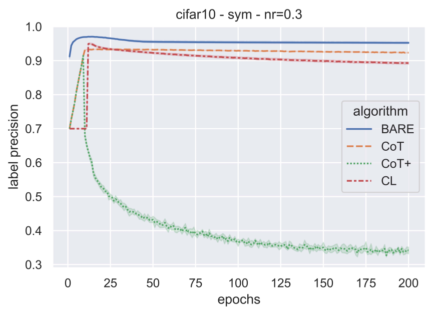


(b)

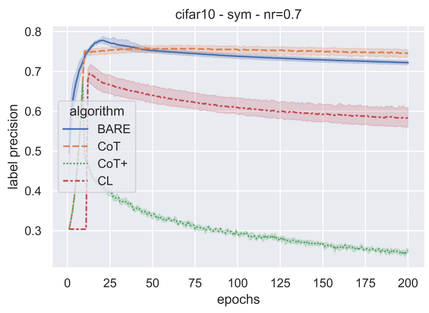


(c)

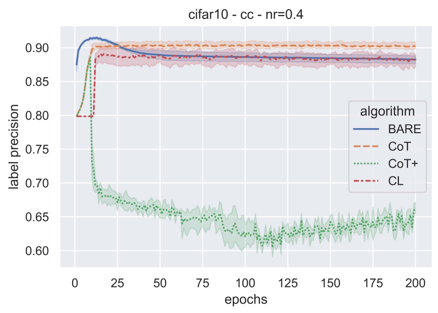
Figure 3.3: Label Precision - MNIST - Symmetric ((a) & (b)) & Class-conditional ((c)) Label Noise



(a)



(b)

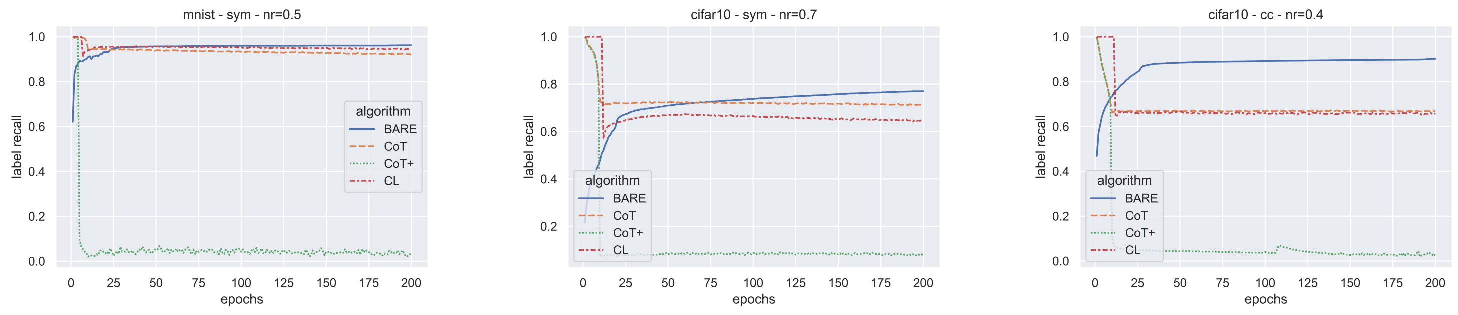


(c)

Figure 3.4: Label Precision - CIFAR10 - Symmetric ((a) & (b)) & Class-conditional ((c)) Label Noise

Efficacy of detecting clean samples

Figure 3.3 and Figure 3.4 show the label precision (across epochs) of the various algorithms on MNIST and CIFAR-10 respectively. One can see from these figures that BARE has comparable or better precision. Thus, compared to other sample selection algorithms, a somewhat higher fraction of samples selected for training by BARE have clean labels. Figure 3.5 show the label recall values for CoT, CoT+, CL, and BARE for MNIST (3.5(a)) and CIFAR-10 (3.5(b) & 3.5(c)). It can be noted that BARE consistently achieves better recall values compared to the baselines. Higher recall values indicate that the algorithm is able to identify clean samples more reliably. This is useful, for example, to employ a label cleaning algorithm on the samples flagged as noisy (i.e., not selected) by BARE. CoT+ selects a fraction of samples where two networks disagree and, hence, after the first few epochs, it selects very few samples (~ 3000) in each epoch. Since these are samples in which the networks disagree, a good fraction of them may have noisy labels. This may be the reason for the poor precision and recall values of CoT+ as seen in these figures. This can be seen from Figure 3.6 as well which shows the fraction of samples chosen by the sample selection algorithms as epochs go by on CIFAR-10 & MNIST dataset. It can be noted that, as noise rate is to be supplied to CoT and CL, they select $1 - \eta = 0.6$ fraction of data with every epoch. Whereas, in case of CoT+, the samples where the networks disagree is small because of the training dynamics and as a result, after a few epochs, it consistently selects very few samples. Since the noise is class-conditional, even though $\eta = 0.4$, the actual amount of label flipping is $\sim 20\%$. And this is why it's interesting to note that BARE leads to an approximate sample selection ratio of 80%.



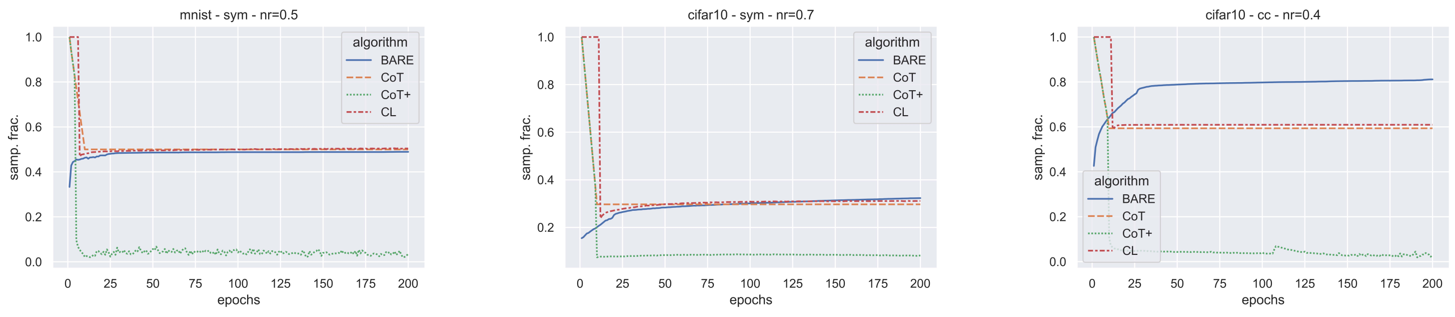
(a) (b) (c)

Figure 3.5: Label Recall - Symmetric ((a) & (b)) & Class-conditional ((c)) Label Noise

Figure 3.6 shows the fraction of samples selected by different algorithms in each epoch. As is evident from these figures, BARE is able to identify higher fraction of clean samples effectively even without the knowledge of noise rates.

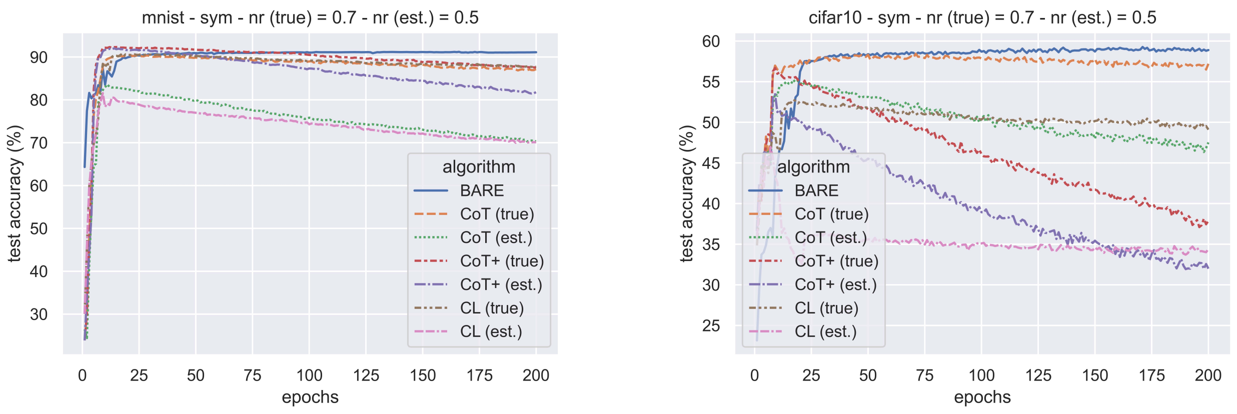
Sensitivity to noise rates

Some of the baselines schemes such as CoT, CoT+, and CL require knowledge of true noise rates beforehand. This information is typically unavailable in practice. One can



(a) (b) (c)

Figure 3.6: (a): Sample fraction values for $\eta = 0.5$ (symmetric noise) on MNIST, (b): Sample fraction values for $\eta = 0.7$ (symmetric noise) on CIFAR-10, (c): sample fraction values for $\eta = 0.4$ (class-conditional noise) on CIFAR-10



(a) (b)

Figure 3.7: Test accuracies when estimated (symmetric) noise rate, $\eta = 0.5$, and true noise rate, $\eta = 0.7$, for (a): MNIST & (b): CIFAR-10

estimate the noise rates but there would be inevitable errors in estimation. Figure 3.7 shows the effect of mis-specification of noise rates for these 3 baselines schemes. As can be seen from these figures, while the algorithms can exhibit robust learning when the true noise rate is known, the performance deteriorates if the estimated noise rate is erroneous. Obviously, BARE does not have this issue because it does not need any information on noise rate.

$$\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0.6 & 0 & 0 & 0 & 0 & 0.3 & 0 & 0.1 \\
0 & 0 & 0 & 0.5 & 0 & 0.1 & 0 & 0 & 0.4 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0.15 & 0.55 & 0.3 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0.35 & 0.55 & 0.10 & 0 & 0 \\
0 & 0.25 & 0 & 0 & 0 & 0 & 0 & 0.5 & 0 & 0.25 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}$$

Arbitrary Noise Matrix for MNIST

Results on Arbitrary Noise Matrix

Earlier, we showed results for special cases of class-conditional noise. There the noise rate is still specifiable by a single number and there is a pre-fixed pairs of classes that can be confused with each other. As explained earlier, we have used this type of noise because that was what was used in literature.

We now provide results for class-conditional noise with an arbitrary, diagonally-dominant noise matrix. We show the noise matrices above. These noise matrices are chosen arbitrarily. However, as can be seen, now in each row more than two entries can be non-zero.

The results obtained with different algorithms are shown in Tables 3.11–3.14. As mentioned earlier, algorithms CoT, CoT+ and CL need knowledge of noise rate. With an arbitrary noise matrix, it is not clear what number can be supplied as the noise rate for these algorithms, even if we know all the noise rates. For these simulations, $\eta = 0.45$ and $\eta = 0.4$ are supplied as the estimated noise rates to CoT, CoT+, and CL baselines for MNIST and CIFAR-10 respectively. In the tables, the best two results are in bold. It can be seen that the proposed algorithm continues to perform well.

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.2 & 0 & 0.7 & 0 & 0 & 0 & 0.1 & 0 & 0 & 0 \\ 0.1 & 0 & 0 & 0.6 & 0 & 0.1 & 0 & 0 & 0.2 & 0 \\ 0 & 0.1 & 0.1 & 0 & 0.7 & 0 & 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0 & 0.1 & 0 & 0.6 & 0 & 0 & 0 & 0.3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0.1 & 0 & 0 & 0 & 0 & 0 & 0.1 & 0 & 0.8 \end{bmatrix}$$

Arbitrary Noise Matrix for CIFAR-10

Table 3.11: Test Accuracy (%) for MNIST - $\eta_{est} = 0.45$ (arbitrary noise matrix)

ALGORITHM	TEST ACCURACY
CoT [37]	**95.3**
CoT+ [122]	93.07
CL [68]	88.41
BARE (Ours)	**95.02**

Table 3.12: Avg. Test Accuracy (last 10 epochs) (%) for MNIST - $\eta_{est} = 0.45$ (arbitrary noise matrix)

ALGORITHM	AVG. TEST ACCURACY (LAST 10 EPOCHS)
CoT [37]	**95.22**
CoT+ [122]	93.08
CL [68]	88.56
BARE (Ours)	**95.03**

Table 3.13: Test Accuracy (%) for CIFAR10 - $\eta_{est} = 0.4$ (arbitrary noise matrix)

Algorithm	Test Accuracy
CoT [37]	71.92
CoT+ [122]	68.56
CL [68]	**72.12**
BARE (Ours)	**76.22**

Table 3.14: Avg. Test Accuracy (last 10 epochs) (%) for CIFAR10 - $\eta_{est} = 0.4$ (arbitrary noise matrix)

Algorithm	Avg. Test Accuracy (last 10 epochs)
CoT [37]	71.86
CoT+ [122]	68.99
CL [68]	**72.27**
BARE (Ours)	**75.96**

3.4 Conclusions

We propose an adaptive, data-dependent sample selection scheme, BARE, for robust learning in the presence of label noise. The algorithm relies on statistics of assigned posterior probabilities of all samples in a mini-batch to select samples from that mini-batch. The mini-batch statistics are used as proxies for determining current state of learning here. Unlike other algorithms in literature, BARE neither needs an extra data set with clean labels nor does it need any knowledge of the noise rates. Further it has no hyperparameters in the selection algorithm. Comparisons with baseline schemes on benchmark datasets show the effectiveness of the proposed algorithm both in terms of performance metrics and computational complexity.

The current algorithms for sample selection in literature rely on heuristics such as cross-training multiple networks or meta-learning of sample weights which is often computationally expensive. They also need knowledge of noise rates or some data with clean labels which may not be easily available. From the results shown here, it seems that relying on the current state of learning via batch statistics alone is helping the proposed algorithm confidently pick out the clean data and ignore the noisy data (without even the need for cross training of two networks). This, combined with the fact that there are no hyperparameters to tune, shows the advantage that BARE can offer for robust learning under label noise.

Chapter 4

Role of Loss Function in Robust Learning

As mentioned in Chapter 2, the problem of learning under label noise has a long history. In the last decade many empirical studies have demonstrated the adverse effects of label noise on learning different classifier models such as neural networks, SVMs, decision trees etc. (See, e.g., [83]). This, coupled with the fact that one needs large labelled data sets for training deep networks, has attracted a lot of attention on learning under label noise.

Recently some studies have shown that deep networks are very effective in interpolating the data. It was empirically shown in [123] that even when the labels in the training data are completely randomly assigned, deep networks are capable of learning parameters that drive training error to zero. This phenomenon has been called memorization. Among other things, this study showed that deep networks are very susceptible to overfitting under label noise and hence motivated a large number of efforts at designing robust learning algorithms. In this chapter we present an empirical study on the ability of a class of loss functions in resisting the memorization of noisy labels.

4.1 Introduction

Memorization in deep networks got a lot of attention recently due to the work of Zhang et al. [123] which showed that standard deep network architectures are highly susceptible to exact interpolation of data. They show that even when one completely randomly alters class labels in the training data, these networks can learn these random labels almost exactly. That is, the SGD based learning algorithm can learn weights such that the training error is close to zero. It is seen that this memorization of the training samples cannot be mitigated through any of the standard regularization techniques such as weight decay or dropout. These results seem to imply that the usual complexity measures of statistical learning theory such as VC-dimension are possibly inadequate to properly understand the generalization performance of deep neural networks. The main motivation for the study is to explore this question of how to explain the generalization performance of deep networks.

There are many further studies that address this issue and study the dynamics of this memorization process. Arpit et al. [6] present an intetresting study that, while confirming the memorization process, presents insights on how this happens. They formulate some characterizations under which the learning dynamics of a network differ for the two cases of learning from real data and random data. Their study suggests that the data itself may be playing a vital role in resisting brute-force memorization by a network. They also found empirical evidence to say that deep networks learn simpler patterns first before starting to memorize the data. Similar features regarding the dynamics of memorization have been reported in [33]. As mentioned in Chapter 3, some of the methods for robust learning under label noise are motivated by this observation that the networks may be learning from clean samples first before overfitting to the noisy ones. Another recent work [25] shows that memorization is necessary for generalization for some types of distributions which has been tested empirically by [26].

All these studies experiment with many scenarios of regularization, other optimization techniques and randomization of data and study their effect on memorization. However, the role that the loss function itself can play in this has not been investigated at all. Given some of the recent results on robustness of risk minimization [30, 73], it would be interesting to investigate whether the variation of training error with learning epochs would be qualitatively different for different loss functions. Motivated by this, here we present some empirical rsults to show that a loss function can also play a significant role in preventing a network from memorizing data.

Neural networks are universal approximators [44] and networks with sufficient parameters have the capacity to exactly represent any finite amount of data [123]. That is, for these networks there **exist** parameter values that can represent any arbitrary function. However, as discussed in [6], what a network learns depends on the parameter values that a gradient-based learning algorithm can reach starting from some random initial parameter values. This learning dynamics is certainly affected, among other factors, by the loss function because the loss function determines the topography of the empirical risk, which is minimized by the learning algorithm. Hence, it would be interesting to investigate whether it is possible to have loss functions that can inherently resist (to some degree) the memorization of data by a network.

Our focus in this chapter is on the role a loss function can play in resisting memorization. Specifically we would be looking at symmetric loss functions. Since memorization is about training error, we will be mainly concerned with that. The other studies on memorization normally use completely random labels. Here we look at the memorization under varying level of label randomization. That is, we experiment with symmetric label noise with different noise rates. This will give us some insights on how certain classes of loss functions resist driving the training error to zero.

We present some experimental results for benchmark datasets, MNIST [59] and CIFAR-10 [56], with labels randomly changed with different probabilities. We see that for varying probabilities of random labelling, networks trained with standard loss functions such as categorical cross entropy (CCE) or mean square error (MSE) exhibit memorization by reaching close to zero training error. We show that keeping everything else in the training algorithm same but changing the loss function to Robust Log Loss (RLL) [57] which is a symmetric loss, results in the network significantly resisting this memorization.

We also present some theoretical justification (using the known properties of these symmetric losses) for the ability of these loss functions to resist memorization. We formally define what can be called *resisting memorization*. Using this, we explain why symmetric loss functions can resist brute-force memorization in these scenarios, thus providing some theoretical justification for the empirically observed performance with RLL.

There are many works that attempt comparative study of different loss functions for classification and regression tasks. With extensive empirical experiments on a variety of data sets, [46] shows that MSE performs better than CCE thus challenging the conventional wisdom of the superiority of CCE loss for classification tasks. It is argued in [20] that CCE is better (compared to MSE) for multi-class settings but a technique is proposed that makes performance of MSE comparable to that of CCE. It is demonstrated in [90, 95] that MSE has comparable or better performance than hinge loss for several tasks. It is shown in [81] that minimizers of risk obtained in case of MSE and hinge loss are the same for overparameterized linear models under certain conditions. These and other similar works compare different loss functions for classification and regression tasks (using clean data) from the point of view of generalization. The material presented here is different from all these and it looks at the role loss functions can play in affecting the degree of memorization in overparameterized networks.

As mentioned earlier, memorization is about training error. However, as we see here, when a loss function results in robustness to memorization, it essentially results in learning from the data that have clean labels. Thus, this study on memorization gives us insights that are useful in designing algorithms for robust learning under label noise.

4.2 Role of loss function in resisting memorization: Empirical Results

We begin by first presenting some empirical results on memorization in some deep networks under different loss functions. We show that the symmetric loss, RLL, shows good ressitance to memorization. We also present empirical results to show that, under RLL, the network is actually trying to learn from data with the uncorrupted labels.

We experiment with two network architectures. One is an Inception-like network architecture (referred to as Inception-Lite in this paper) which is same as that used in [123] for demonstrating memorization in deep networks. The second is ResNet-32 (and ResNet-18 for MNIST) architecture as used in [101]. The details of the Inception-Lite network architecture are provided in Figure 5.1.

We present results with three loss functions. Two are the standard loss functions used with neural networks, namely, CCE and MSE, and the third is a symmetric loss, viz.

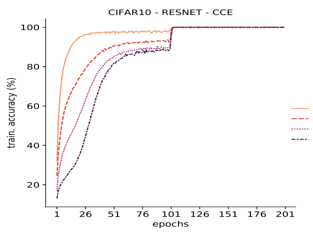
RLL. Since we are considering classification problems, for all the networks we assume a softmax output layer. For an input, $\mathbf{x}$, let $\mathbf{f}(\mathbf{x})$ denote the vector output of the network with components $f_i(\mathbf{x})$. When $\mathbf{x}$ belongs to class k, the label would be the one-hot vector $\mathbf{e}_k$ where $e_{kk} = 1$ and $e_{kj} = 0$, $\forall j \neq k$. Let K denote the number of classes. With this notation, the three loss functions can be defined as follows:

$$\begin{aligned}
\mathcal{L}_{CCE}(\mathbf{f}(\mathbf{x}), \mathbf{e}_k) &= -\sum_i e_{ki} \, \log\left(f_i(\mathbf{x})\right) \, = -\log(f_k(\mathbf{x})) \\
\mathcal{L}_{MSE}(\mathbf{f}(\mathbf{x}), \mathbf{e}_k) &= \sum_i \left(f_i(\mathbf{x}) - e_{ki}\right)^2 \\
\mathcal{L}_{RLL}(\mathbf{f}(\mathbf{x}), \mathbf{e}_k) &= \log\left(\frac{\alpha+1}{\alpha}\right) - \log(\alpha + f_k(\mathbf{x})) + \sum_{j \neq k} \frac{1}{K-1} \log(\alpha + f_j(\mathbf{x}))
\end{aligned}$$

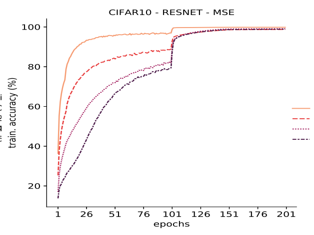
where $\alpha > 0$ is a parameter of the RLL.

We can get some insights on behaviour of RLL versus CCE as follows: When $\mathbf{x}$ is in class-k, $f_k(\mathbf{x})$ is the posterior probability assigned to class-k by the network. If this is high, then the CCE loss, which is $-\log(f_k(\mathbf{x}))$, is low. However, the CCE loss is unbounded because, in principle, there can be network functions such that $f_k(x)$ can be arbitrarily small. Disregarding the constant term, the RLL takes $-\log(\alpha + f_k(\mathbf{x})) + \sum_{j \neq k} \frac{1}{K-1} \log(\alpha + f_j(\mathbf{x}))$ as its value. Since we are using $\log(\alpha + f_j(\mathbf{x}))$ rather than $\log(f_j(\mathbf{x}))$, the loss is now bounded. More importantly, the loss is essentially determined through a kind of comparison of the posterior probability assigned to class-k by the network against the average probability assigned to all other classes. (The constant term in RLL is there only to ensure that the loss is non-negative). As we shall see, this gives some amount of robustness in the risk minimization resulting in RLL exhibiting good resistance to memorization.

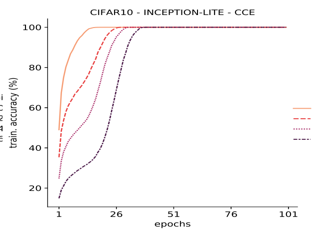
We train all the networks to minimize empirical risk (with each of the loss functions). We employ mini-batch based stochastic gradient descent (SGD) for Inception-Lite & ResNet-32 (for CIFAR-10) and Adam [55] for ResNet-18 (for MNIST). For Inception-Lite, we use a constant step-size of 0.01 in each epoch which is reduced by a factor of 0.95 after each epoch for 100 epochs whereas a constant step-size of 0.1 is used for ResNet-32 which is reduced by a factor of 0.1 after 100 and 150 epochs. ResNet-32 & ResNet-18 are trained for 200 epochs. The ResNet-18 is trained with a step-size of 0.001. Inception-Lite



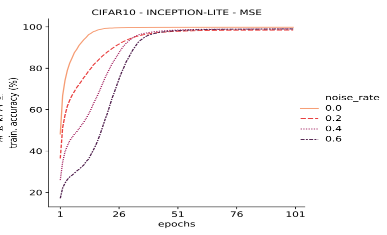
(a) CCE



(b) MSE



(c) CCE



(d) MSE

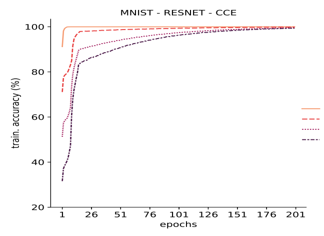
Figure 4.1: Training set accuracies for ResNet-32 ((a) & (b)) & Inception-Lite ((c) & (d)) trained on CIFAR-10 with CCE and MSE losses for for $\eta \in \{0., 0.2, 0.4, 0.6\}$

is trained for 100 epochs because the training accuracies saturate by then. Inception-Lite and ResNet-18 are trained without weight decay whereas ResNet-32 is trained with a weight decay of 0.0001. The implementations use NumPy [40], scikit-learn [88], and PyTorch [85]. All the codes for these experiments are available here: `https://github.com/dbp1994/masters_thesis_codes/tree/main/memorization_and_overparam`

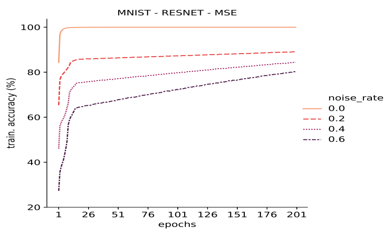
CIFAR-10 and MNIST benchmark datasets are used for the experiments. As explained earlier, we study the memorization by the networks through randomly altering the class labels in the training set. For this, independently for each sample, we retain the original label with probability $(1 - \eta)$ and change it with probability η. When the label is changed, it is changed to one of the other classes with equal probability. We experiment with $\eta = 0, 0.2, 0.4$, and 0.6. (Note that $\eta = 0$ corresponds to the clean or original training data). That is, we are introducing symmetric label noise with different noise rates.

Figure 4.1 shows the training accuracies achieved with ResNet-32 and Inception-Lite when we train the network with CCE & MSE for various values of η on CIFAR-10. As can be seen from the figure, the training accuracy goes to 100% for these two networks with both the loss functions, CCE and MSE. With increasing noise rate, it takes more epochs to reach 100% training accuracy. But, even with noise rate of 60% the networks are able to perfectly fit the noisy labels.

Figure 4.2 shows training accuracies of ResNet-18 with CCE and MSE for MNIST. Even though this is a smaller network, when we train it with CCE, we still get 100% training accuracy irrespective of the noise rate showing that the network perfectly fits

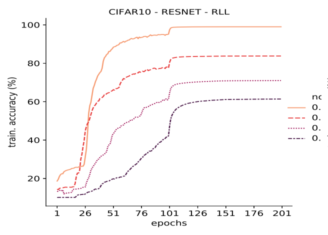


(a) CCE

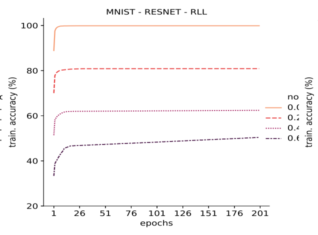


(b) MSE

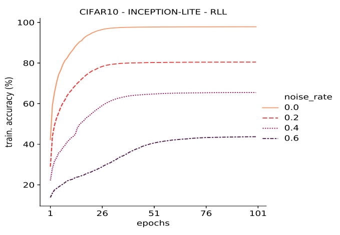
Figure 4.2: Training set accuracies for ResNet-18 trained on MNIST with CCE and MSE losses for different levels of label noise



(a) ResNet-32



(b) ResNet-18



(c) Inception-Lite

Figure 4.3: Training set accuracies for networks trained on CIFAR-10 ((a) & (c)) & MNIST ((b)) with RLL for $\eta \in \{0., 0.2, 0.4, 0.6\}$

the random labels. Under MSE, for noisy data, the training accuracy has not become 100% by 200 epochs. However, as is clearly seen from the trend in the graph, the training acuracies under different noise rates are increasing, coming together and going towards 100%. Our experimental results are consistent with the results reported in [6, 123] for CCE loss. (All studies in memorization considered only CCE loss). (Note that [123] show training set performances only for $\eta = 1$ and do not experiment with varying levels of noise as was done here.) Note that at $\eta = 0.2$, 80% of training samples of a class are correctly labelled and hence would contain the patterns that the network would have learnt when trained with clean data. However, the network ends up learning a function that can exactly reproduce the training set labels. This seems to indicate that with these loss functions the topography of the empirical risk function is such that the learning dynamics takes the network to a point that fits the random labels exactly.

These results may be contrasted with those presented in Figure 4.3 which are obtained when the same networks are trained with RLL for different values of η on MNIST & CIFAR-10. As can be seen from the figures, the training set accuracy achieved by RLL for non-zero values of η is always well below that achieved on clean data. Also the accuracy achieved is lower for a higher value of η. The training set accuracies saturate and stay constant after about 100 epochs showing that even if we train the network for another 100 epochs we cannot make it fit the random labels exactly. This shows that the network does not blindly learn to reproduce the training set labels. This is significant because this shows that when we keep everything else same and change only the loss function, the learning dynamics now seem to be able to resist brute-force memorization. Also, for $\eta = 0.2$ and $\eta = 0.4$, the difference in training-accuracy on clean and noisy data is almost equal to the noise-rate thus suggesting that this loss function seems to be able to disregard data that are wrongly labelled.

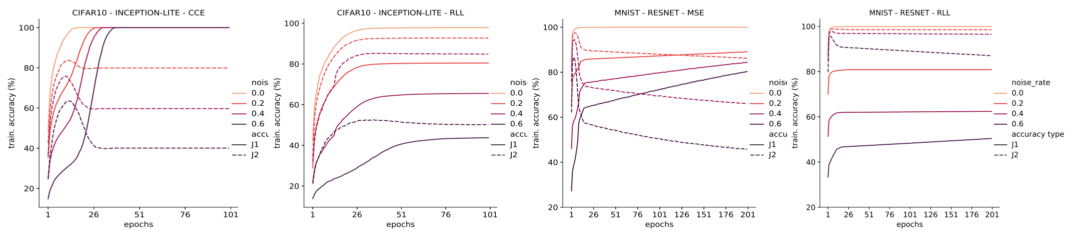
We now take a closer look to understand the kind of classifier learnt by RLL under noisy data. We use the same notation as in the earlier chapters. Let $\{\mathbf{x}_i, y_i^{cl}\}_{i=1}^{\ell}$ denote the original training data (with 'clean' labels) and let $\{\mathbf{x}_i, y_i\}_{i=1}^{\ell}$ denote the noisy or randomly-labelled data given to the learning algorithm. Let $f(\mathbf{x})$ be the function represented by the network and let $h(\mathbf{x}) = \arg\max_i f_i(\mathbf{x})$ denote the actual class label predicted by the network for $\mathbf{x}$. Then the training accuracy, say J_1, is defined by

$$J_1 = \frac{1}{\ell} \sum_{i=1}^{\ell} I_{[h(\mathbf{x}_i) = y_i)]}$$

where I_A is indicator of A. This is, the accuracy defined with respect to the labels as given in the training set. We define another accuracy, J_2, by

$$J_2 = \frac{1}{\ell} \sum_{i=1}^{\ell} I_{[h(\mathbf{x}_i) = y_i^{cl})]}$$

J_2 is the accuracy, of the same network h, but with respect to original, uncorrupted or clean training set. This accuracy indicates how well the network, learned with randomly-altered labels, would be able to reproduce the original clean labels of the training data. Note that J_2 cannot be considered as the test accuracy. We are measuring accuracy of predictions on the same $\mathbf{x}_i$ as in the training set but with respect to the unaltered labels.



(a) CCE (b) RLL (c) MSE (d) RLL

Figure 4.4: J_1 and J_2 accuracies for Inception-Lite ((a) & (b)) & ResNet-18 ((c) & (d)) trained on CIFAR-10 and MNIST resp. for $\eta \in \{0., 0.2, 0.4, 0.6\}$ (Solid lines show J_1 accuracy; dashed lines show J_2 accuracy)

We show in Figure 4.4 the accuracies J_1 and J_2 for networks learned with the different loss functions for different values of η. As can be seen from Figures 4.4(a) & 4.4(c) for networks trained with CCE and MSE losses, the final J_2 accuracy (dashed line) is always below the final J_1 accuracy (solid line). This shows that these networks are better at predicting the training labels (which are randomly altered ones) of the training data rather than the original labels of the training data. This is as expected because, as seen earlier, the training accuracy, which is equal to J_1, is close to 100%. However, for networks learned with RLL (Figure 4.4(b) & 4.4(d)), it is the J_2 accuracy (dashed line) that is always higher than the J_1 accuracy (solid line). As a matter of fact, for $\eta = 0.2, 0.4$, the J_2 accuracy of the networks learned using RLL is close to the training accuracy achieved with clean data. This suggests that this loss function is able to disregard the randomly altered labels and help the network learn a classifier that it would have learned with clean data.

There is another interesting point about this figure. The figure shows how the J_1 and J_2 accuracies evolve with epochs. As can be seen from the figure, the networks learned using CCE with noisy data seemed to have initially tried to learn the patterns and thus the J_2 accuracy is higher in the early epochs. But eventually the network 'flips' and fits the random labels in training data. However, this 'flip' never happens for networks trained using RLL; through all the epochs, the J_2 accuracy stays higher.

All the empirical results presented in this section amply demonstrate that a loss function can play a significant role in mitigating the memorization effect observed with deep neural networks. In the next section, we present some theoretical analysis that explains, to some extent, the results presented in this section.

4.3 Robustness of Symmetric Loss Functions

In [123], for networks learned using training data with random labels, the accuracy obtained on part of the original data is taken as test error for the purpose of discussing the generalization abilities. However, this may be somewhat of an inaccurate nomenclature. Normally the test error is error on new data but drawn from the same distribution as that from which training data is drawn. However, we can formalize the notion of resisting memorization in a manner similar to the way we defined inherent robustness to label noise in Chapter 1 (Equation 1.11). We start with such a definition and later slightly modify it to capture the notion of memorization which is about training error.

For this section, for simplicity of notation, we assume class labels, y_i, take values in $\mathcal{Y} = \{1, \cdots, K\}$ rather than being one-hot vectors. Except for this, we follow the same notation as earlier. Let $S = \{\mathbf{x}_i, y_i^{cl}\}_{i=1}^{\ell}$ be the original training data and we assume it is drawn *iid* according to a distribution $\mathcal{D}$. The training data with randomly altered labels is denoted by $S_\eta = \{\mathbf{x}_i, y_i\}_{i=1}^{\ell}$, and we assume this is corrupted by symmetric noise with rate η. Thus, for each i,

$$y_i = \begin{cases} y_i^{cl} & \text{with probability } \; 1-\eta \\ j \in \mathcal{Y} - \{y_i^{cl}\} & \text{with probability } \; \frac{\eta}{K-1} \end{cases} \tag{4.1}$$

We denote the distribution from which S_η is drawn as $\mathcal{D}_\eta$. and it is related to $\mathcal{D}$ as given above.

When one is investigating memorization of random labels, one is using training data drawn according to distribution $\mathcal{D}_\eta$ but is interested in test error according to distribution $\mathcal{D}$. As a matter of fact, we want the learnt network to do well on data only from $\mathcal{D}$; we do not want it to learn distribution $\mathcal{D}_\eta$. Hence, we can formally define being robust to memorization in the same way as being robust to label noise.

Let h and h_η denote the classifier function (network) learned by an algorithm when given S and S_η as training data, respectively. We can say that an algorithm **resists**

memorization if

$$\text{Prob}_{(X,y)\sim\mathcal{D}}[h(X)=y] = \text{Prob}_{(X,y)\sim\mathcal{D}}[h_\eta(X)=y] \tag{4.2}$$

. What this means is that the accuracy on the original distribution achieved by the two networks – one learnt with noisy data and the other with the original clean data – is the same.

As mentioned earlier, memorization is about training error. We only want to look at the training error with and without label noise. If training error with noisy labels also goes to zero, we want to call it memorization. So, resisting memorization should be about learning original labels (rather than blindly memorize the randomly altered labels) for the training data patterns. However, the probabilities above are with respect to underlying distribution from which training data is drawn. In this sense, the above definition does not capture the notion of memorization well.

We can modify the definition by simply taking $\mathcal{D}$ to be the empirical distribution determined by training set S. Then the above definition is satisfactory because we are now talking strictly about training error. Thus the final definition we use for memorization is as follows.

Let h and h_η denote the classifier function (network) learned by an algorithm when given S and S_η as training data, respectively. We can say that an algorithm **resists memorization** if

$$\frac{1}{m}\sum_{i=1}^{m}\mathbb{I}_{\{h(\mathbf{x}_i)=y_i^{cl}\}} = \frac{1}{m}\sum_{i=1}^{m}\mathbb{I}_{\{h_\eta(\mathbf{x}_i)=y_i^{cl}\}} \tag{4.3}$$

The LHS of Equation (4.3) is what we called J_1 accuracy under the original or clean data. The RHS of Equation (4.3) is what we called J_2 accuracy. So, a learning algorithm ideally resists memorization if the J_2 accuracy achieved by it under randomly altered labels is equal to J_1 accuracy on original clean data. As we saw in the previous section, performance of empirical risk minimization with RLL is reasonably close to this ideal behaviour.

For the rest of this section we take $\mathcal{D}$ to be the empirical distribution defined by S.

Remark: Note that when we define resistance to memorization using the empirical distribution, the definition is dependent on a training set, S. In the literature, memorization is explored with respect to specific data sets such as MNIST or CIFAR. Hence, it may be alright to ask whether an algorithm resists memorization of a specific data set. To be more rigorous, we can define an algorithm to be resisting memorization if it can resist memorization of any training set, S, drawn from $\mathcal{D}$. But that may be too stringent.

We next explore the connection between symmetric losses and memorization. As mentioned earlier, RLL is a symmetric loss [57]. Let us recall definition of symmetric loss.

Definition: A loss function L is called **symmetric** if $\exists\, C \in \mathbb{R}_+$ such that

$$\sum_{j=1}^{K} \mathcal{L}(f(\mathbf{x}), j) = C, \quad \forall f, \forall \mathbf{x}$$

That is, given any network (or function) f and any input $\mathbf{x}$, if we sum the loss values over all class labels, it should give the same constant. Recall from Chapter 1 that a symmetric loss is bounded in the sense that $\mathcal{L}(f(\mathbf{x}), j) \leq C, \;\; \forall f, \; \forall \mathbf{x}, \; \forall j$.

The following theorem gives some theoretical justification for the observed performance of RLL. The theorem is applicable for any general distributions $\mathcal{D}$ and $\mathcal{D}_\eta$ under symmetric label noise. We can use it to understand resistance to memorization by taking $\mathcal{D}$ to be the empirical distribution defined by S and $\mathcal{D}_\eta$ to be the distribution related to $\mathcal{D}$ as defined by Equation (4.1). (We denote by $(\mathbf{x}, y^{cl}_{\mathbf{x}})$ the random variable with distribution $\mathcal{D}$ and $(\mathbf{x}, y_{\mathbf{x}})$ the random variable with distribution $\mathcal{D}_\eta$).

Theorem 1 Let $\mathcal{L}$ be a symmetric loss, $\mathcal{D}$ and $\mathcal{D}_\eta$ be as defined above. Assume $\eta < \frac{K-1}{K}$. The risk of h over $\mathcal{D}$ and over $\mathcal{D}_\eta$ is $R_{\mathcal{L}}(h) = \mathbb{E}_{\mathcal{D}}[\mathcal{L}(h(\mathbf{x}), y^{cl}_{\mathbf{x}})]$ and $R^\eta_{\mathcal{L}}(h) = \mathbb{E}_{\mathcal{D}_\eta}[\mathcal{L}(h(\mathbf{x}), y_{\mathbf{x}})]$ respectively. Then, given any two classifiers h_1 and h_2, if $R_{\mathcal{L}}(h_1) < R_{\mathcal{L}}(h_2)$, then $R^\eta_{\mathcal{L}}(h_1) < R^\eta_{\mathcal{L}}(h_2)$ and vice versa.

Proof (This follows easily from the proof of Theorem 1 in [30].) Given the way the randomized labels are generated, we have

$$\begin{aligned} R^\eta_{\mathcal{L}}(h) &= \mathbb{E}_{\mathbf{x}, y_{\mathbf{x}}} \mathcal{L}(h(\mathbf{x}), y_{\mathbf{x}}) \\ &= \mathbb{E}_{\mathbf{x}} \mathbb{E}_{y^{cl}_{\mathbf{x}}|\mathbf{x}} \mathbb{E}_{y_{\mathbf{x}}|\mathbf{x}, y^{cl}_{\mathbf{x}}} \mathcal{L}(h(\mathbf{x}), y_{\mathbf{x}}) \end{aligned}$$

$$
\begin{aligned}
&= \mathbb{E}_{\mathbf{x}}\mathbb{E}_{y_{\mathbf{x}}^{cl}|\mathbf{x}}\left[(1-\eta)\mathcal{L}(h(\mathbf{x}), y_{\mathbf{x}}^{cl}) + \frac{\eta}{K-1}\sum_{i\neq y_{\mathbf{x}}^{cl}}\mathcal{L}(h(\mathbf{x}), i)\right] \\
&= (1-\eta)R_{\mathcal{L}}(h) + \frac{\eta}{K-1}(C - R_{\mathcal{L}}(h)) \\
&= \frac{C\eta}{K-1} + \left(1 - \frac{\eta K}{K-1}\right)R_{\mathcal{L}}(h)
\end{aligned}
$$

where C is the constant in the symmetry condition on the loss function and K is the number of classes. Since $\eta < \frac{K-1}{K}$, we have $(1 - \frac{\eta K}{K-1}) > 0$. Hence, the above shows that whenever $R_{\mathcal{L}}(h_1) < R_{\mathcal{L}}(h_2)$, we get $R_{\mathcal{L}}^{\eta}(h_1) < R_{\mathcal{L}}^{\eta}(h_2)$ and vice versa. This completes the proof. ■

Theorem 1 shows that the symmetric loss maintains the risk ranking of different networks regardless of random flipping of labels (as long as $\eta < \frac{K-1}{K}$). Since we are taking the distributions to be empirical distributions of the training data, this implies that, fixing a training set, any local minimum of empirical risk under randomly flipped labels would also be a local minimum of risk under original labels if the loss function is symmetric. This explains the empirical results presented in the previous section regarding the ability of RLL to resist memorization.

This theorems tells us why empirical risk minimization with a symmetric loss such as RLL would resist memorization. However, there is one caveat. When we are given data with randomly altered labels the algorithm goes to a local minimum of empirical risk under noisy distribution. The theorem above guarantees that this is also a local minimum of empirical risk under clean training data too. However, when we do empirical risk minimization with clean data we may reach some other (and better) local minimum. That is the reason why though RLL will prevent minimization to a good degree, it does not have the ideal behavious (where J_2 accuracy would be same as J_1 accuracy on clean data). If our algorithm is guaranteed to take us to a global minimizer of empirical risk (with a symmetric loss function) then the above theorem implies we will realize the ideal behaviour.

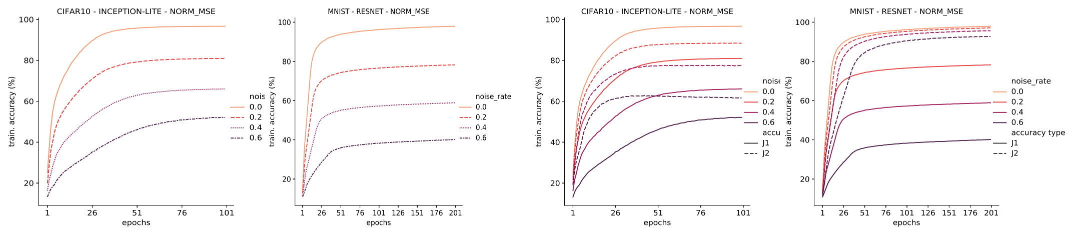
There are other losses that satisfy the symmetry condition, e.g., 0–1 loss, mean absolute value of error (MAE), etc.

It is easy to verify that neither CCE nor MSE satisfy the symmetry condition. Though the symmetry of loss is only a sufficient condition for robustness, this may provide an explanation of why risk minimization with these losses results in memorization.

As mentioned earlier, the symmetry condition implies that the loss function is bounded. That is, for any arbitrary network, f, and for any $\mathbf{x}$, we must have $\mathcal{L}(f(\mathbf{x}), i) < C$, $\forall i \in \mathcal{Y}$ where C is a fixed constant. Given a bounded loss function we can satisfy the symmetry condition by 'normalizing' it. Given a bounded loss $\mathcal{L}$, define $\bar{\mathcal{L}}$ by

$$\bar{\mathcal{L}}(f(\mathbf{x}), j) = \frac{\mathcal{L}(f(\mathbf{x}), j)}{\sum_s \mathcal{L}(f(\mathbf{x}), s)}$$

It is easy to see that $\bar{L}$ satisfies the symmetry condition. As mentioned earlier, CCE loss is unbounded and hence normalization would not turn it into a symmetric loss. However, we can normalize MSE loss.



(a) Norm. MSE (b) Norm. MSE (c) Norm. MSE (d) Norm. MSE

Figure 4.5: Train. accuracy and J_1 & J_2 accuracies for Inception-Lite ((a) & (c)) & ResNet-18 ((b) & (d)) trained on CIFAR-10 and MNIST resp. for $\eta \in \{0., 0.2, 0.4, 0.6\}$ (Solid lines show J_1 accuracy; dashed lines show J_2 accuracy)

In Figure 4.5 we show results obtained using normalized MSE. Once we normalize MSE, it no longer fits the data with random labels perfectly; the training accuracy now saturates at a value well below 100% and it behaves more like RLL now.

The empirical results presented in the previous section adequately demonstrate that the loss function can play a crucial role in mitigating the tendency of deep networks to memorize the training samples. The analysis presented here provides an explanation for this ability of RLL to resist such memorization. As a mater of fact, if the loss function is symmetric it would have such robustness and we can normalize a bounded loss to satisfy the symmetry condition.

4.4 Conclusions

Many recent studies have shown that overparameterized deep networks seem to be capable of perfectly fitting even randomly-labelled data. This phenomenon of memorization in deep networks has received a lot of attention because it raises important questions on how to understand generalization abilities of deep networks. In this chapter we have shown through empirical studies that changing the loss function alone can significantly change the memorization in such deep networks. We showed this happens when the loss function is symmetric. We have provided some theoretical analysis to explain the empirical results. The results presented here suggest that choice of loss function can play a critical role in overfitting by deep networks and hence in the design of algorithms for robust learning. We also feel that the study presented here can be useful in getting more insights in the understanding of generalization abilities of deep networks.

Chapter 5

Conclusions

This thesis presents some investigations on the problem of robust supervised learning under label noise using neural networks. In most supervised learning applications one needs to handle label noise. This is because large scale labelled data sets are obtained through mechanisms such as crowd-sourcing. Random labelling errors due to human errors, subjective biases, and so on also makes label noise invetible in many cases. In Chapter 1 we discussed the relevance of studying this problem and what robust learning under label noise means.

In Chapter 2 we presented a detailed review of the existing methods for tackling label noise most of which are heuristics-based and do not have a guarantee for robust learning. We focus on one of the recent and popular approaches, namely, *sample reweighting*, in more detail in Chapter 3.

In Chapter 3 we proposed a new sample selection strategy for robust learning under label noise which we called BAtch REweighting (BARE). It can be viewed as an adaptive curriculum based learning and focuses on the current state of learning, in a given mini-batch, for identifying the noisily-labelled data in it. For this, the statistics of posterior probabilities of all samples in a mini-batch are used to compute the threshold for sample selection. This yields an adaptive curriculum where the sample selection is naturally tied to state of learning. Our algorithm does not need any clean validation data, needs no knowledge at all of the noise rates and also does not have any hyperparameters. We empirically demonstrated the effectiveness of our algorithm on three benchmark datasets, namely, MNIST and CIFAR-10 and Clothing-1M. We showed that, in terms of robustness, it is as good as or better than all the state of art methods for sample reweighting or sample selection. We also showed that it is much more efficient in terms of time taken for learning.

In Chapter 4 we study 'memorization' in neural network. This is the phenomenon of the learning algorithm converging to weights that result in zero (or close to zero) training error even when training data is randomly labelled. Memorization is seen to be a cause for poor performance of overapameterized neural networks to generalize well on unseen data if the training data has label noise. None of the studies on memorization investigated the role played by the loss function. Since memorization is about finding certain kinds of local minima in the empirical risk, the loss function should be playing a crucial role here. In Chapter 4 of the thesis we presented an empirical study, using data sets MNIST and CIFAR-10, on how the choice of loss function affects memorization. We showed that a special class of loss functions, namely, symmetric losses, can resist memorization to a good degree. We formally defined what 'resisting memorization' means and used some known properties of symmetric loss functions to provide some theoretical justifications for the empirical results.

Future Work

Given the effectiveness of BARE in selecting clean samples and providing robustness against label noise as seen in the empirical results, a potential future direction is to better understand how and why the batch statistics help the algorithm reliably select good samples. Another potential future direction is to identify other and better proxies for current state of learning apart from batch statistics that offer such robustness. Another interesting direction in which the algorithm can be explored further is to see whether we can add label cleaning on top of the sample selection given by BARE.

In the context of 'memorization', a theoretical understanding of the effect different classes of loss functions can have on the optimization landscape is needed to better understand the role of loss functions. This would also be helpful in getting a theory of robust learning under label noise. This can be a future direction. It will also be interesting to understand how do symmetric losses automatically identify noisy data (in as much as they do not overfit for them). This would help find new mechanisms for identification of noisy data.

Appendix

The Inception-Lite architecture used for experiments in Chapter 4 is described here[1]:

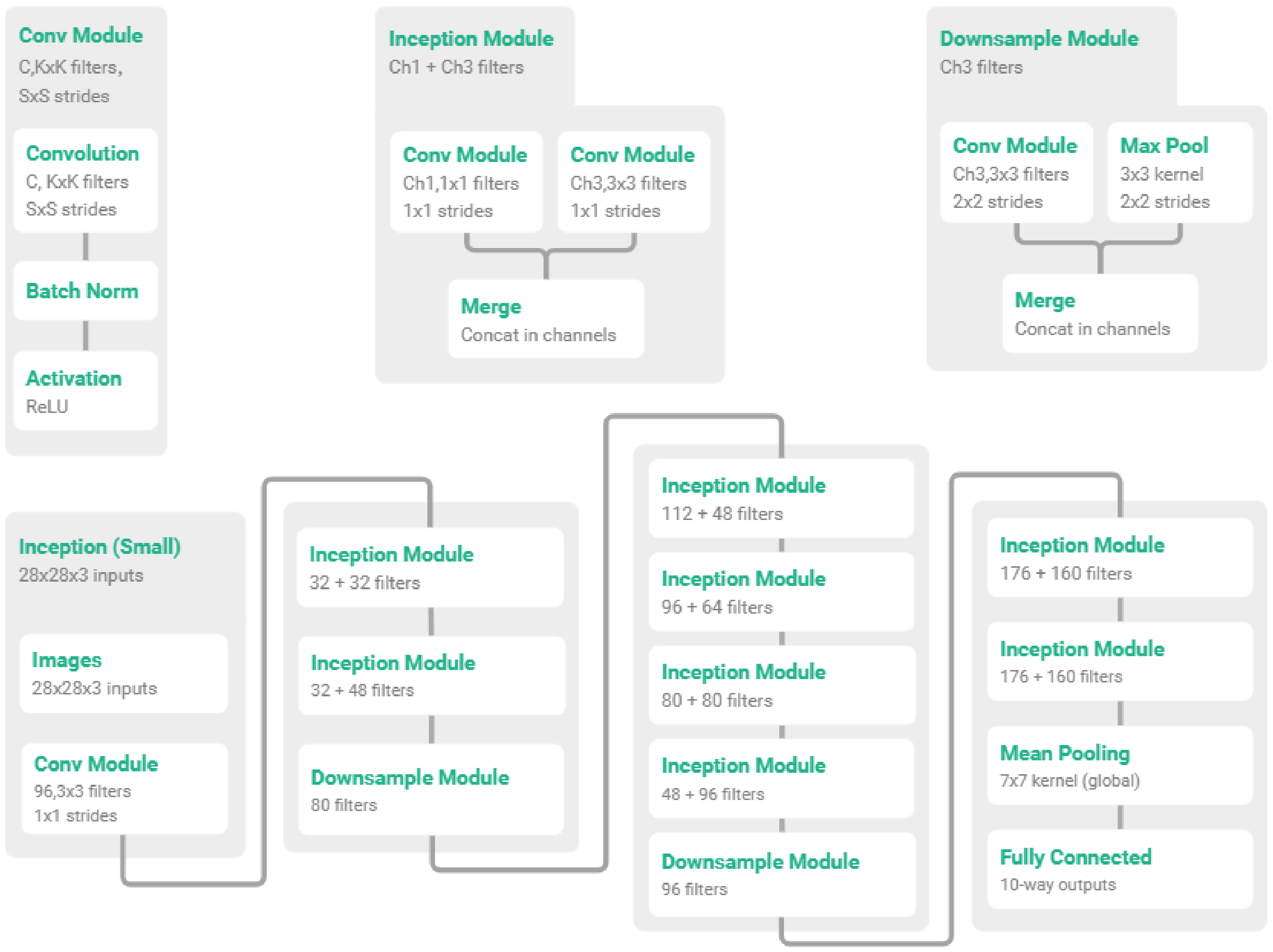


Figure 5.1: Inception-Lite

[1]Source: https://arxiv.org/pdf/1611.03530.pdf

References

[1] Guillaume Alain, Alex Lamb, Chinnadhurai Sankar, Aaron Courville, and Yoshua Bengio. Variance reduction in sgd by distributed importance sampling. *arXiv preprint arXiv:1511.06481*, 2015.

[2] Kamal M Ali and Michael J Pazzani. Error reduction through learning multiple descriptions. *Machine learning*, 24(3):173–202, 1996.

[3] Dana Angluin and Philip Laird. Learning from noisy examples. *Machine Learning*, 2(4):343–370, 1988.

[4] Eric Arazo, Diego Ortego, Paul Albert, Noel O'Connor, and Kevin McGuinness. Unsupervised label noise modeling and loss correction. In *International Conference on Machine Learning*, pages 312–321. PMLR, 2019.

[5] Eric Arazo, Diego Ortego, Paul Albert, Noel O'Connor, and Kevin McGuinness. Unsupervised label noise modeling and loss correction. In *International Conference on Machine Learning*, pages 312–321. PMLR, 2019.

[6] Devansh Arpit, Stanislaw Jastrzebski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, et al. A closer look at memorization in deep networks. In *International Conference on Machine Learning*, pages 233–242. PMLR, 2017.

[7] Mihai Badoiu and Kenneth L Clarkson. Smaller core-sets for balls. In *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 801–802. Society for Industrial and Applied Mathematics, 2003.

[8] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48, 2009.

[9] David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin Raffel. Mixmatch: A holistic approach to semi-supervised learning. *arXiv preprint arXiv:1905.02249*, 2019.

[10] Battista Biggio, Blaine Nelson, and Pavel Laskov. Support vector machines under adversarial label noise. In *Asian conference on machine learning*, pages 97–112, 2011.

[11] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100, 1998.

[12] Carla E Brodley and Mark A Friedl. Identifying mislabeled training data. *Journal of artificial intelligence research*, 11:131–167, 1999.

[13] Haw-Shiuan Chang, Erik Learned-Miller, and Andrew McCallum. Active bias: Training more accurate neural networks by emphasizing high variance samples. In *Advances in Neural Information Processing Systems*, pages 1002–1012, 2017.

[14] Nontawat Charoenphakdee, Jongyeong Lee, and Masashi Sugiyama. On symmetric losses for learning from corrupted labels. In *International Conference on Machine Learning*, pages 961–970. PMLR, 2019.

[15] Pengfei Chen, Guangyong Chen, Junjie Ye, jingwei zhao, and Pheng-Ann Heng. Noise against noise: stochastic label noise helps combat inherent label noise. In *International Conference on Learning Representations*, 2021.

[16] Pengfei Chen, Ben Ben Liao, Guangyong Chen, and Shengyu Zhang. Understanding and utilizing deep neural networks trained with noisy labels. In *International Conference on Machine Learning*, pages 1062–1070, 2019.

[17] Pengfei Chen, Junjie Ye, Guangyong Chen, Jingwei Zhao, and Pheng-Ann Heng. Beyond class-conditional assumption: A primary attempt to combat instance-dependent label noise. *arXiv preprint arXiv:2012.05458*, 2020.

[18] Luis Daza and Edgar Acuna. An algorithm for detecting noise on supervised classification. In *Proceedings of WCECS-07, the 1st World Conference on Engineering and Computer Science*, pages 701–706, 2007.

[19] Mostafa Dehghani, Aliaksei Severyn, Sascha Rothe, and Jaap Kamps. Learning to learn from weak supervision by full supervision. *arXiv preprint arXiv:1711.11383*, 2017.

[20] A. Demirkaya, J. Chen, and S. Oymak. Exploring the role of loss functions in multiclass classification. In *2020 54th Annual Conference on Information Sciences and Systems (CISS)*, pages 1–5, 2020.

[21] Thomas G Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine learning*, 40(2):139–157, 2000.

[22] Guiguang Ding, Yuchen Guo, Kai Chen, Chaoqun Chu, Jungong Han, and Qionghai Dai. DECODE: Deep confidence network for robust image classification. *IEEE Transactions on Image Processing*, 28(8):3752–3765, 2019.

[23] Nan Ding and SVN Vishwanathan. t-logistic regression. In *Advances in Neural Information Processing Systems*, pages 514–522, 2010.

[24] Amir Erfan Eshratifar, David Eigen, and Massoud Pedram. Gradient agreement as an optimization objective for meta-learning. *arXiv preprint arXiv:1810.08178*, 2018.

[25] Vitaly Feldman. Does learning require memorization? a short tale about a long tail. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 954–959, 2020.

[26] Vitaly Feldman and Chiyuan Zhang. What neural networks memorize and why: Discovering the long tail via influence estimation. *Advances in Neural Information Processing Systems*, 33, 2020.

[27] Lei Feng, Senlin Shu, Zhuoyi Lin, Fengmao Lv, Li Li, and Bo An. Can cross entropy loss be robust to label noise. In *Proceedings of the 29th International Joint Conferences on Artificial Intelligence*, pages 2206–2212, 2020.

[28] Shai Fine, Ran Gilad-bachrach, Shahar Mendelson, and Naftali Tishby. Noise tolerant learnability via the dual learning problem. 1999.

[29] Jinyang Gao, H. V. Jagadish, and Beng Chin Ooi. Active sampler: Light-weight accelerator for complex data analytics at scale. *CoRR*, abs/1512.03880, 2015.

[30] Aritra Ghosh, Himanshu Kumar, and PS Sastry. Robust loss functions under label noise for deep neural networks. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 1919–1925, 2017.

[31] Aritra Ghosh, Naresh Manwani, and PS Sastry. Making risk minimization tolerant to label noise. *Neurocomputing*, 160:93–107, 2015.

[32] Yves Grandvalet, Yoshua Bengio, et al. Semi-supervised learning by entropy minimization. In *CAP*, pages 281–296, 2005.

[33] Jindong Gu and Volker Tresp. Neural network memorization dissection, 2019.

[34] I. Guyon, N. Matić, and V. Vapnik. Discovering informative patterns and data cleaning. In *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*, page 145–156, 1994.

[35] Bo Han, Gang Niu, Xingrui Yu, Quanming Yao, Miao Xu, Ivor Tsang, and Masashi Sugiyama. SIGUA: Forgetting may make learning with noisy labels more robust. In *International Conference on Machine Learning*, pages 4006–4016. PMLR, 2020.

[36] Bo Han, Jiangchao Yao, Gang Niu, Mingyuan Zhou, Ivor Tsang, Ya Zhang, and Masashi Sugiyama. Masking: A new perspective of noisy supervision. *arXiv preprint arXiv:1805.08193*, 2018.

[37] Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor Tsang, and Masashi Sugiyama. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *Advances in neural information processing systems*, pages 8527–8537, 2018.

[38] Jiangfan Han, Ping Luo, and Xiaogang Wang. Deep self-learning from noisy labels. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5138–5147, 2019.

[39] Sariel Har-Peled, Dan Roth, and Dav Zimak. Maximum margin coresets for active and noise tolerant learning. In *IJCAI*, pages 836–841, 2007.

[40] Charles R Harris, K Jarrod Millman, Stéfan J van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe,

Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, Sept. 2020.

[41] Hrayr Harutyunyan, Kyle Reing, Greg Ver Steeg, and Aram Galstyan. Improving generalization by controlling label-noise information in neural network weights. In *International Conference on Machine Learning*, pages 4071–4081. PMLR, 2020.

[42] Ryuichiro Hataya and Hideki Nakayama. Lol: Learning to optimize loss switching under label noise. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 3621–3625. IEEE, 2019.

[43] Dan Hendrycks, Mantas Mazeika, Duncan Wilson, and Kevin Gimpel. Using trusted data to train deep networks on labels corrupted by severe noise. In *Advances in neural information processing systems*, pages 10456–10465, 2018.

[44] Kurt Hornik, Maxwell Stinchcombe, Halbert White, et al. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[45] Wei Hu, Zhiyuan Li, and Dingli Yu. Simple and effective regularization methods for training on noisily labeled data with generalization guarantee. In *International Conference on Learning Representations*, 2020.

[46] Like Hui and Mikhail Belkin. Evaluation of neural architectures trained with square loss vs cross-entropy in classification tasks, 2020.

[47] Simon Jenni and Paolo Favaro. Deep bilevel learning. In *Proceedings of the European conference on computer vision (ECCV)*, pages 618–633, 2018.

[48] Lu Jiang, Deyu Meng, Shoou-I Yu, Zhenzhong Lan, Shiguang Shan, and Alexander Hauptmann. Self-paced learning with diversity. In *Advances in Neural Information Processing Systems*, pages 2078–2086, 2014.

[49] Lu Jiang, Deyu Meng, Qian Zhao, Shiguang Shan, and Alexander Hauptmann. Self-paced curriculum learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

[50] Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *International Conference on Machine Learning*, pages 2304–2313. PMLR, 2018.

[51] Wenxin Jiang. Some theoretical aspects of boosting in the presence of noisy data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, page 234–241, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[52] George H John. Robust decision trees: Removing outliers from databases. In *KDD*, volume 95, pages 174–179, 1995.

[53] Amitava Karmaker and Stephen Kwek. A boosting approach to remove class label noise. *International Journal of Hybrid Intelligent Systems*, 3(3):169–177, 2006.

[54] Youngdong Kim, Junho Yim, Juseung Yun, and Junmo Kim. NLNL: Negative learning for noisy labels. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 101–110, 2019.

[55] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[56] Alex Krizhevsky. *Learning Multiple Layers of Features from Tiny Images.* PhD thesis, University of Toronto, 2009.

[57] H. Kumar and P. S. Sastry. Robust loss functions for learning multi-class classifiers. In *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 687–692, 2018.

[58] M Pawan Kumar, Benjamin Packer, and Daphne Koller. Self-paced learning for latent variable models. In *Proceedings of the 23rd International Conference on Neural Information Processing Systems-Volume 1*, pages 1189–1197, 2010.

[59] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[60] Kuang-Huei Lee, Xiaodong He, Lei Zhang, and Linjun Yang. Cleannet: Transfer learning for scalable image classifier training with label noise. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5447–5456, 2018.

[61] Junnan Li, Richard Socher, and Steven C.H. Hoi. Dividemix: Learning with noisy labels as semi-supervised learning. In *International Conference on Learning Representations*, 2020.

[62] Jia Li, Yafei Song, Jianfeng Zhu, Lele Cheng, Ying Su, Lin Ye, Pengcheng Yuan, and Shumin Han. Learning from large-scale noisy web data with ubiquitous reweighting for image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[63] Junnan Li, Yongkang Wong, Qi Zhao, and Mohan S Kankanhalli. Learning to learn from noisy labeled data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5051–5059, 2019.

[64] Or Litany and Daniel Freedman. SOSELETO: A unified approach to transfer learning and training with noisy labels, 2019.

[65] Sheng Liu, Jonathan Niles-Weed, Narges Razavian, and Carlos Fernandez-Granda. Early-learning regularization prevents memorization of noisy labels. *Advances in Neural Information Processing Systems*, 33, 2020.

[66] Tongliang Liu and Dacheng Tao. Classification with noisy labels by importance reweighting. *IEEE Transactions on pattern analysis and machine intelligence*, 38(3):447–461, 2015.

[67] Philip M Long and Rocco A Servedio. Random classification noise defeats all convex potential boosters. *Machine learning*, 78(3):287–304, 2010.

[68] Yueming Lyu and Ivor W. Tsang. Curriculum loss: Robust learning and generalization against label corruption. In *International Conference on Learning Representations*, 2020.

[69] Xingjun Ma, Hanxun Huang, Yisen Wang, Simone Romano, Sarah Erfani, and James Bailey. Normalized loss functions for deep learning with noisy labels. In *International Conference on Machine Learning*, pages 6543–6553. PMLR, 2020.

[70] Xingjun Ma, Yisen Wang, Michael E Houle, Shuo Zhou, Sarah Erfani, Shutao Xia, Sudanthi Wijewickrema, and James Bailey. Dimensionality-driven learning with noisy labels. In *International Conference on Machine Learning*, pages 3355–3364. PMLR, 2018.

[71] Xingjun Ma, Yisen Wang, Michael E. Houle, Shuo Zhou, Sarah Erfani, Shutao Xia, Sudanthi Wijewickrema, and James Bailey. Dimensionality-driven learning

with noisy labels. In *Proceedings of the 35th International Conference on Machine Learning*, pages 3355–3364, 2018.

[72] Eran Malach and Shai Shalev-Shwartz. Decoupling" when to update" from" how to update". In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 961–971, 2017.

[73] Naresh Manwani and PS Sastry. Noise tolerance under risk minimization. *IEEE transactions on cybernetics*, 43(3):1146–1151, 2013.

[74] Hamed Masnadi-Shirazi and Nuno Vasconcelos. On the design of loss functions for classification: theory, robustness to outliers, and savageboost. In *Advances in neural information processing systems*, pages 1049–1056, 2009.

[75] Ross A McDonald, David J Hand, and Idris A Eckley. An empirical comparison of three boosting algorithms on real data sets with artificial class noise. In *International Workshop on Multiple Classifier Systems*, pages 35–44. Springer, 2003.

[76] Prem Melville, Nishit Shah, Lilyana Mihalkova, and Raymond J Mooney. Experiments on ensembles with missing and noisy data. In *International Workshop on Multiple Classifier Systems*, pages 293–302. Springer, 2004.

[77] Deyu Meng, Qian Zhao, and Lu Jiang. What objective does self-paced learning indeed optimize? *arXiv preprint arXiv:1511.06049*, 2015.

[78] Baharan Mirzasoleiman, Kaidi Cao, and Jure Leskovec. Coresets for robust training of neural networks against noisy labels. *arXiv preprint arXiv:2011.07451*, 2020.

[79] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1979–1993, 2018.

[80] Volodymyr Mnih and Geoffrey E Hinton. Learning to label aerial images from noisy data. In *Proceedings of the 29th International conference on machine learning (ICML-12)*, pages 567–574, 2012.

[81] Vidya Muthukumar, Adhyyan Narang, Vignesh Subramanian, Mikhail Belkin, Daniel Hsu, and Anant Sahai. Classification vs regression in overparameterized regimes: Does the loss function matter?, 2020.

[82] Nagarajan Natarajan, Inderjit S Dhillon, Pradeep K Ravikumar, and Ambuj Tewari. Learning with noisy labels. In *Advances in neural information processing systems*, pages 1196–1204, 2013.

[83] David F Nettleton, Albert Orriols-Puig, and Albert Fornells. A study of the effect of different types of noise on the precision of supervised learning techniques. *Artificial Intelligence Review*, 33(4):275–306, 2010.

[84] David Opitz and Richard Maclin. Popular ensemble methods: An empirical study. *Journal of artificial intelligence research*, 11:169–198, 1999.

[85] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alche-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[86] Giorgio Patrini, Alessandro Rozza, Aditya Krishna Menon, Richard Nock, and Lizhen 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 (CVPR)*, July 2017.

[87] Mykola Pechenizkiy, Alexey Tsymbal, Seppo Puuronen, and Oleksandr 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)*, pages 708–713. IEEE, 2006.

[88] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[89] Te Pi, Xi Li, Zhongfei Zhang, Deyu Meng, Fei Wu, Jun Xiao, and Yueting Zhuang. Self-paced boost learning for classification. In *IJCAI*, pages 1932–1938, 2016.

[90] Qichao Que and Mikhail Belkin. Back to the future: Radial basis function networks revisited. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pages 1375–1383, Cadiz, Spain, 09–11 May 2016. PMLR.

[91] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[92] Scott E Reed, Honglak Lee, Dragomir Anguelov, Christian Szegedy, Dumitru Erhan, and Andrew Rabinovich. Training deep neural networks on noisy labels with bootstrapping. In *ICLR (Workshop)*, 2015.

[93] Mark D Reid and Robert C Williamson. Composite binary losses. *The Journal of Machine Learning Research*, 11:2387–2422, 2010.

[94] Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4334–4343, 2018.

[95] Ryan Michael Rifkin. *Everything old is new again: a fresh look at historical approaches in machine learning*. PhD thesis, Massachussets Insitute of Technology, 2002.

[96] PS Sastry, GD Nagendra, and Naresh Manwani. A team of continuous-action learning automata for noise-tolerant learning of half-spaces. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 40(1):19–28, 2009.

[97] Bernhard Scholkopf, John Platt, and Thomas Hofmann. *Correcting Sample Selection Bias by Unlabeled Data*, pages 601–608. 2007.

[98] Clayton Scott. A rate of convergence for mixture proportion estimation, with application to learning from noisy labels. In *Artificial Intelligence and Statistics*, pages 838–846, 2015.

[99] Clayton Scott, Gilles Blanchard, and Gregory Handy. Classification with asymmetric label noise: Consistency and maximal denoising. In *Conference On Learning Theory*, pages 489–511, 2013.

[100] Yanyao Shen and Sujay Sanghavi. Learning with bad training data via iterative

trimmed loss minimization. In *International Conference on Machine Learning*, pages 5739–5748, 2019.

[101] Jun Shu, Qi Xie, Lixuan Yi, Qian Zhao, Sanping Zhou, Zongben Xu, and Deyu Meng. Meta-weight-net: Learning an explicit mapping for sample weighting. In *Advances in Neural Information Processing Systems*, pages 1919–1930, 2019.

[102] Hwanjun Song, Minseok Kim, and Jae-Gil Lee. Selfie: Refurbishing unclean samples for robust deep learning. In *ICML*, pages 5907–5915, 2019.

[103] Guillaume Stempfel and Liva Ralaivola. Learning svms from sloppily labeled data. In *International conference on artificial neural networks*, pages 884–893. Springer, 2009.

[104] Daiki Tanaka, Daiki Ikami, Toshihiko Yamasaki, and Kiyoharu Aizawa. Joint optimization framework for learning with noisy labels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5552–5560, 2018.

[105] Kagan Tumer and Joydeep Ghosh. Analysis of decision boundaries in linearly combined neural classifiers. *Pattern Recognition*, 29(2):341–348, 1996.

[106] L. G. Valiant. Learning disjunction of conjunctions. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence - Volume 1*, page 560–566, San Francisco, CA, USA, 1985. Morgan Kaufmann Publishers Inc.

[107] Brendan Van Rooyen, Aditya Menon, and Robert C Williamson. Learning with symmetric label noise: The importance of being unhinged. In *Advances in Neural Information Processing Systems*, pages 10–18, 2015.

[108] Andreas Veit, Neil Alldrin, Gal Chechik, Ivan Krasin, Abhinav Gupta, and Serge Belongie. Learning from noisy large-scale datasets with minimal supervision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 839–847, 2017.

[109] Yisen Wang, Xingjun Ma, Zaiyi Chen, Yuan Luo, Jinfeng Yi, and James Bailey. Symmetric cross entropy for robust learning with noisy labels. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 322–330, 2019.

[110] Zhen Wang, Guosheng Hu, and Qinghua Hu. Training noise-robust deep neural networks via meta-learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4524–4533, 2020.

[111] Hongxin Wei, Lei Feng, Xiangyu Chen, and Bo An. Combating noisy labels by agreement: A joint training method with co-regularization. 2020.

[112] Jiaheng Wei and Yang Liu. When optimizing \$f\$-divergence is robust with label noise. In *International Conference on Learning Representations*, 2021.

[113] Sanford Weisberg. *Applied linear regression*, volume 528, page 194. John Wiley & Sons, 2005.

[114] Pengxiang Wu, Songzhu Zheng, Mayank Goswami, Dimitris Metaxas, and Chao Chen. A topological filter for learning with label noise. *Advances in Neural Information Processing Systems*, 33, 2020.

[115] Xiaobo Xia, Tongliang Liu, Bo Han, Chen Gong, Nannan Wang, Zongyuan Ge, and Yi Chang. Robust early-learning: Hindering the memorization of noisy labels. In *International Conference on Learning Representations*, 2021.

[116] Xiaobo Xia, Tongliang Liu, Bo Han, Nannan Wang, Mingming Gong, Haifeng Liu, Gang Niu, Dacheng Tao, and Masashi Sugiyama. Part-dependent label noise: Towards instance-dependent label noise. *Advances in Neural Information Processing Systems*, 33, 2020.

[117] Xiaobo Xia, Tongliang Liu, Nannan Wang, Bo Han, Chen Gong, Gang Niu, and Masashi Sugiyama. Are anchor points really indispensable in label-noise learning? *arXiv preprint arXiv:1906.00189*, 2019.

[118] Tong Xiao, Tian Xia, Yi Yang, Chang Huang, and Xiaogang Wang. Learning from massive noisy labeled data for image classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2691–2699, 2015.

[119] Linli Xu, Koby Crammer, and Dale Schuurmans. Robust support vector machine training via convex outlier ablation. In *AAAI*, volume 6, pages 536–542, 2006.

[120] Quanming Yao, Hansi Yang, Bo Han, Gang Niu, and James Tin-Yau Kwok. Searching to exploit memorization effect in learning with noisy labels. In *Proceedings of*

the 37th International Conference on Machine Learning, volume 119 of *Proceedings of Machine Learning Research*, pages 10789–10798. PMLR, 2020.

[121] Kun Yi and Jianxin Wu. Probabilistic end-to-end noise correction for learning with noisy labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7017–7025, 2019.

[122] Xingrui Yu, Bo Han, Jiangchao Yao, Gang Niu, Ivor Tsang, and Masashi Sugiyama. How does disagreement help generalization against label corruption? In *Proceedings of the 36th International Conference on Machine Learning*, pages 7164–7173, 2019.

[123] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

[124] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

[125] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

[126] Jian Zhang and Yiming Yang. Robustness of regularized linear classification methods in text categorization. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 190–197, 2003.

[127] Yikai Zhang, Songzhu Zheng, Pengxiang Wu, Mayank Goswami, and Chao Chen. Learning with feature dependent label noise: a progressive approach. *arXiv preprint arXiv:2103.07756*, 2021.

[128] Zhilu Zhang and Mert R Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. *arXiv preprint arXiv:1805.07836*, 2018.

[129] Zijin Zhao, Lingyang Chu, Dacheng Tao, and Jian Pei. Classification with label noise: a markov chain sampling framework. *Data Mining and Knowledge Discovery*, 33(5):1468–1504, 2019.

[130] Guoqing Zheng, Ahmed Hassan Awadallah, and Susan Dumais. Meta label correction for noisy label learning. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence*, 2021.

[131] Songzhu Zheng, Pengxiang Wu, Aman Goswami, Mayank Goswami, Dimitris Metaxas, and Chao Chen. Error-bounded correction of noisy labels. In *International Conference on Machine Learning*, pages 11447–11457. PMLR, 2020.

[132] Songzhu Zheng, Pengxiang Wu, Aman Goswami, Mayank Goswami, Dimitris Metaxas, and Chao Chen. Error-bounded correction of noisy labels. In *International Conference on Machine Learning*, pages 11447–11457. PMLR, 2020.

[133] Tianyi Zhou and Jeff Bilmes. Minimax curriculum learning: Machine teaching with desirable difficulties and scheduled diversity. In *International Conference on Learning Representations*, 2018.

[134] Xingquan Zhu, Peng Zhang, Xindong Wu, Dan He, Chengqi Zhang, and Yong Shi. Cleansing noisy data streams. In *2008 Eighth IEEE International Conference on Data Mining*, pages 1139–1144. IEEE, 2008.