LEARNING FROM NOISY LABELS

A Project Report

submitted by

AJAY SHRIDHAR JOSHI

in partial fulfilment of the requirements

for the award of the degree of

BACHELOR OF TECHNOLOGY

AND

MASTER OF TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY MADRAS.

June 5, 2020

THESIS CERTIFICATE

This is to certify that the thesis titled **LEARNING FROM NOISY LABELS**, submitted by **Ajay S Joshi**, to the Indian Institute of Technology, Madras, for the award of the degree of **Bachelor of Technology and Master of Technology**, is a bona fide record of the research work done by him under our supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Prof. Anurag Mittal
Research Guide
Professor
Dept. of CSE
IIT Madras, 600036

Place: Chennai

Date: June 5, 2020

ACKNOWLEDGEMENTS

Firstly, I would like to thank my guide Dr. Anurag Mittal for his constant support and guidance, without which this project would not have been possible. I am grateful to Arulkumar Subramaniam for helping me progress in the project in several ways and for all the discussions we had. My family has supported me constantly, and I would like to express my gratitude towards them. I am also grateful to my friends Sarthak, Prasanna, Milind and many more for their help. Finally, I would like to thank all the members of the Computer Vision Lab for their constant support.

ABSTRACT

The availability of large-scale labeled datasets is one of the crucial requirements for supervised machine learning. Such datasets have paved the way for progress in areas like computer vision, NLP, speech processing, etc. Though such large-scale labeled datasets are important, labeling is resource-intensive and still produces wrong labels to some extent. Thus, there is a need for supervised learning systems which are robust to incorrect labels.

In this work, we approach the problem of noisy labels, by designing training algorithms to filter out data with incorrect labels using basic ideas in machine learning as well as insights from the DNN training process. The contributions of this work are the following:

1. We identify multiple characteristics differentiating examples with clean labels from those with noisy ones.

2. We propose 2 novel training algorithms, which can filter out data with noisy labels, one of which performs better than the state-of-the-art for MNIST and CIFAR-10 datasets.

3. We make novel observations regarding variation of model performance with model complexity in the DNN training process, which helps in learning from noisy labels.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

IITM	Indian Institute of Technology, Madras
DNN	Deep Neural Network
CAM	Class Activation Map
GAP	Global Average Pooling
ERM	Empirical Risk Minimization

CHAPTER 1

Introduction

In this chapter, we give an overview of the problem of learning from noisy labels and provide the motivation behind this work. Then we briefly discuss the contributions of this work in the area and the organization of the rest of the thesis.

1.1 Overview and Problem Statement

In this section, we give an overview of the area of learning from noisy labels and discuss the research question that this work tries to solve.

1.1.1 Classification

Classification is an important problem in machine learning which has been widely studied. It has applications in areas like computer vision, in tasks like image classification and video classification where images/videos are classified into pre-defined categories/ classes. It also has applications in speech processing and natural language processing, where problems like spam detection, sentiment analysis, etc. are formulated as classification problems. Classification problems are approached with supervised machine learning algorithms, which require that the data used is labeled, i.e. consists of (input, output) pairs, and in the case of classification, consists of (input, class label) pairs. Classification tasks are formulated with a definition of the set of classes, and an assignment of each input data point to exactly one of the classes. The performance of supervised learning algorithms on classification tasks is highly sensitive to the quality of the labels in training data and the performance degrades considerably in the presence of noise in labels, i.e. when the labels provided are incorrect to some extent.

1.1.2 Label Noise

Labeled datasets are very crucial to the development of classification algorithms. The process of building such a labeled dataset often requires high amount of resources, either in the form of time, capital or domain expertise for labeling. Labeling requires the definition of appropriate classes and the assignment of each input data point to exactly one of the classes. Small labeled datasets can usually be labeled by domain experts, in which case the amount of error in labeling is fairly low. But, using small amount of data usually gives limited performance, and for building a high performance classification system, large labeled datasets are very crucial. State-of-the-art classification systems for tasks like image classification, video classification, etc. are deep-learning systems, whose performance usually increases with increasing quantity of data. So, large-scale labeled datasets are crucial for high-performing classification systems.

Due to the sheer size of the data to be labeled, labeling using small number of domain experts is infeasible, and alternative approaches have to be used. These approaches include crowd-sourcing technologies like Amazon Mechanical Turk, automated labeling of web-data using image tags or querying web-data using search engines and using query keywords as the labels, etc.. Crowd-sourcing technologies introduce errors in the labeling due to non-expert labeling by the crowd. Automated labeling is inherently prone to incorrect labeling as the tags relevant to images, or search results are imperfect. Data poisoning by an adversary is another way through which labels can be corrupted. Thus, we have provided a brief overview of some of the important reasons for the introduction of errors in labels.

Frénay and Verleysen (2013) provides a taxonomy of label noise which can be described as follows:

1. NCAR(Noise Completely at Random): Errors in labels are independent of any other factor. eg: Figures 1.1, 1.2
2. NAR(Noise at Random): Errors in labels are only dependent on the true class, and independent of input data features. eg: Figure 1.3
3. NNAR(Noise Not at Random): Errors in labels are dependent on the true label as well as the input data features as well.

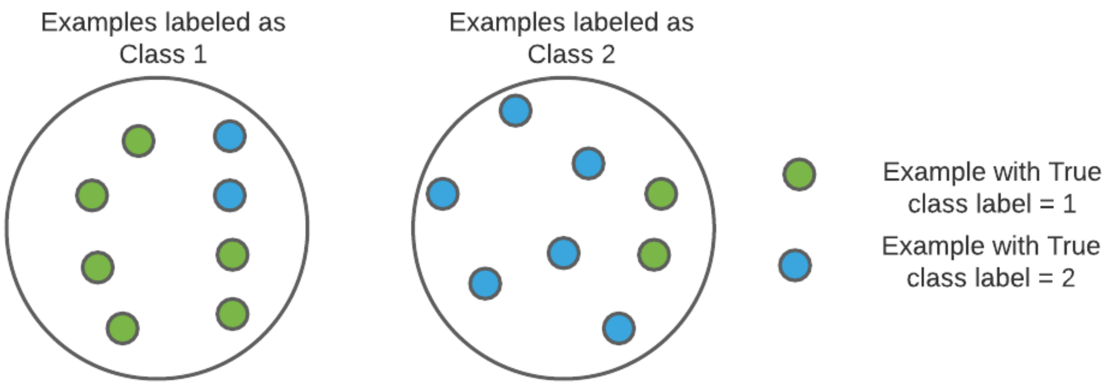

Examples labeled as Class 1
Examples labeled as Class 2
Example with True class label = 1
Example with True class label = 2

Figure 1.1: 25% corruption of labels in class 1 and class 2 both in a binary classification setting. This represents noise which is NCAR, assuming this data sample to be representative of the population.

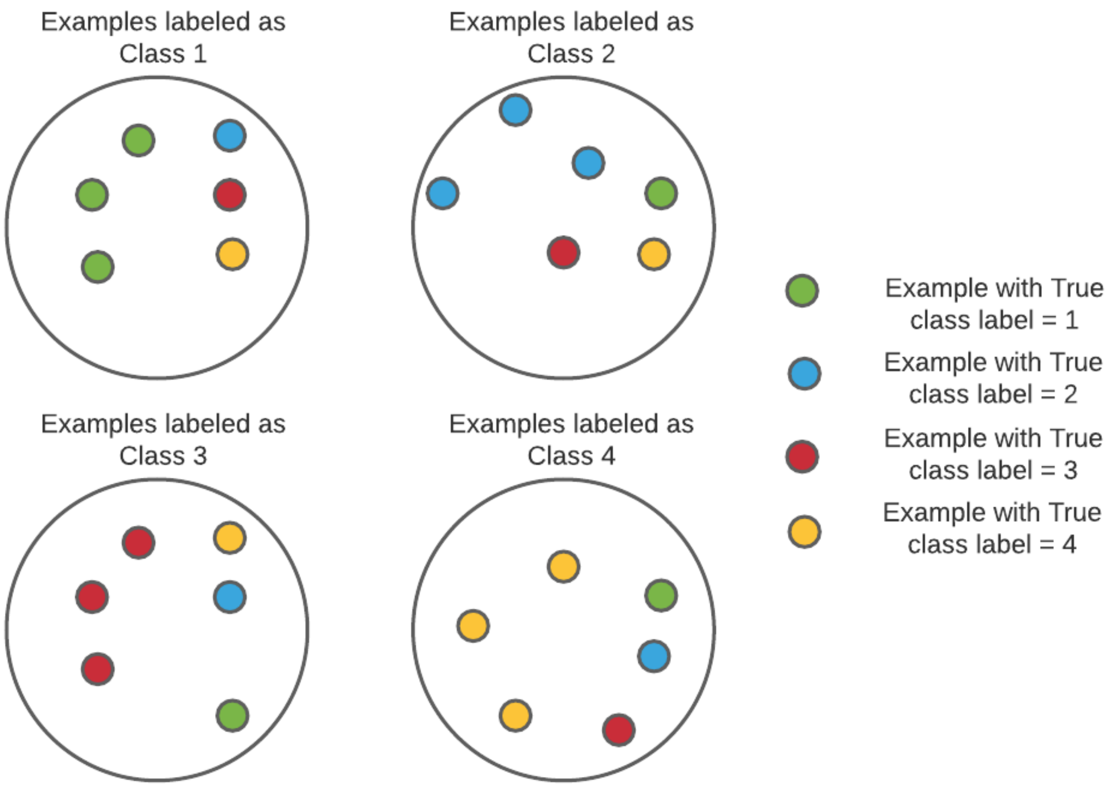

Examples labeled as Class 1
Examples labeled as Class 2
Examples labeled as Class 3
Examples labeled as Class 4
Example with True class label = 1
Example with True class label = 2
Example with True class label = 3
Example with True class label = 4

Figure 1.2: 50% of examples have corrupt labels for all classes in a multi-class classification setting. This represents noise which is NCAR, assuming this data sample to be representative of the population.

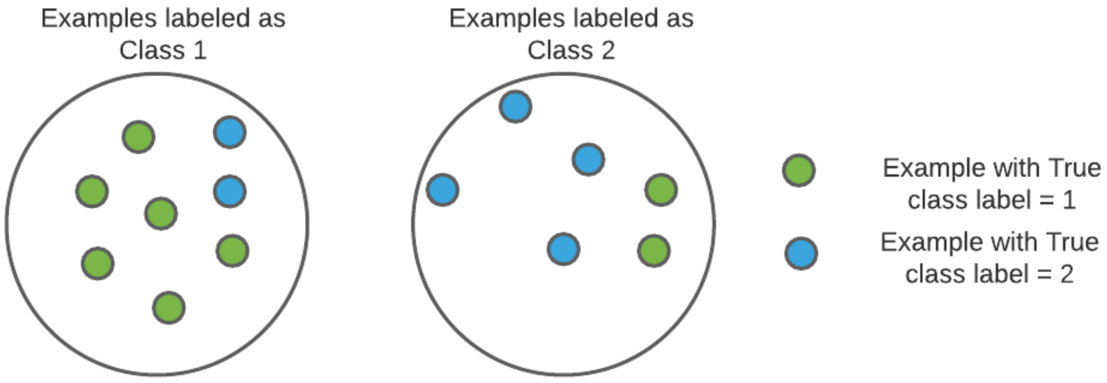

Examples labeled as Class 1
Examples labeled as Class 2
Example with True class label = 1
Example with True class label = 2

Figure 1.3: Percentage of examples having corrupt labels is not same for each class in a binary classification setting. This represents noise which is NAR, assuming this data sample to be representative of the population.

In datasets labeled by automatic systems, the noise observed is usually NAR or NNAR, eg: Clothing-1M Xiao *et al.* (2015). The possible reason for this might be that, only specific classes are hard to distinguish between according to the class definitions, for eg: the classes knitwear and sweater in the Clothing-1M dataset are closely related to each other and thus the chances of confusion between these 2 classes is higher than between the classes T-shirt and shawl. Such confusion could arise in non-expert manual annotators, or in automatic labeling systems.

Label noise degrades performance of supervised machine learning algorithms. DNNs are overparameterized and have high representation power and thus, can represent the distribution corresponding to the noisy labels Zhang *et al.* (2016). So, they suffer severe performance degradation when trained on datasets with noisy labels. Thus, it is important to develop label noise-robust algorithms to train DNNs.

1.1.3 Problem Statement

The objective of this work is to develop algorithms to make machine learning models robust to incorrect labels in labeled datasets.

There has been very little work on generalization of solutions across types and amount of label noise, which makes the problem hard. Also, theoretical results in this area are very limited and thus of limited practical relevance.

1.2 Motivation

The availability of large-scale labeled datasets is a crucial necessity for machine learning models to provide good performance. Though the labels in such datasets are usually obtained through crowdsourcing, which are verified by multiple human annotators, the probability of errors in labels still remains considerable. This can deteriorate the performance of machine learning models trained on such datasets.

The size of datasets which can be labeled manually remains limited, given the huge amount of resources required for labeling. An alternative to this is to learn from web data, in which data is obtained from the web and the labels can be automatically generated using the input queries, considering the queries as some form of annotation of the data, eg: Images obtained from a search engine in response to a query can be labeled using the input query. But, a limitation of this method is that the labels generated are inherently noisy, eg: YFCC100M Thomee *et al.* (2016), Clothing-1M Xiao *et al.* (2015), Food-101N Lee *et al.* (2018).

Thus, the ability to train machine learning models in the presence of noisy labels in datasets is of huge practical importance, as it can mitigate the effects of noise in labels on performance, and it also makes effective utilization of large-scale internet data possible.

1.3 Contributions

We hypothesize that there are inherent patterns in data, which the model can learn in the initial stages of training, irrespective of the labels being corrupted. These learnt patterns will help the model subsequently learn better patterns and perform better, by iteratively decreasing the fraction of examples with incorrect labels.

We start by identifying the characteristics that distinguish examples with correct labels from those with incorrect labels. We formulate algorithms that use these characteristics in order to separate data with clean labels from the data with noisy labels.

We identify characteristics based on Arpit *et al.* (2017), that a neural network learns simpler patterns in initial stages and thus the data with correct labels(clean data) which has some inherent pattern is easier to classify correctly than the data with incorrect labels(noisy data). We visualize class activation maps(CAMs)Zhou *et al.* (2016) which are more focused on the right areas for clean data, but do not exhibit any pattern for noisy data. We develop an algorithm based on the above characteristics which iteratively filters out noisy data by reweighting examples.

We observe that controlling regularization can control the difficulty of the examples that a model can classify correctly, and apply this idea to develop an iterative algorithm to filter out noisy data. Also, we make novel observations regarding how the performance of a model varies with its complexity in the presence of noise. These novel observations help us improve the performance of the aforementioned algorithms.

We experiment on the MNIST, CIFAR10 and CIFAR100 datasets and experiment with 3 different types of noise, using the accuracy on a clean test dataset as the performance metric. By comparing our results with recent work like Han *et al.* (2018); Patrini *et al.* (2017); Jiang *et al.* (2017), we show an improvement in performance in every dataset for all noise cases. Analyzing the final weights given to examples reveals that our algorithm identifies clean and noisy examples successfully with high accuracy, which strengthens our results.

The major contributions of this work are as follows:

1. We identify characteristics that distinguish examples with correct labels from examples with incorrect labels.
2. Using these characteristics, we formulate 2 algorithms that iteratively filter out data with noisy labels.
3. We improve the above algorithms by making novel observations pertaining to variation of model performance with model complexity in the presence of noise using 2 measures of model complexity.

1.4 Organization of the Thesis

The rest of the thesis is organized as follows:

1. Chapter 2 provides an overview of the important concepts used in the rest of the thesis, and places our work with respect to the literature in this area.

2. Chapter 3 details our contribution by detailing the characteristics differentiating examples with correct and incorrect labels, describing our novel observations and ties all of the elements together into algorithms.

3. Chapter 4 gives details regarding the experimental setup, experiments performed, their results and presents a detailed analysis and comparison with existing methods.

4. Chapter 5 gives a summary of our work and our core contributions. We also provide directions and problems for future work.

CHAPTER 2

Background and Related Work

In this chapter, we give a brief overview of concepts used in the rest of the thesis. We also discuss prior work, and place our work in relation with the prior work in this area.

2.1 Background

This section builds up the background and concepts necessary for understanding the ideas discussed in the rest of the thesis. We discuss notions of model complexity from different angles and the idea of class activation maps and their significance.

2.1.1 Model Complexity

The representation power of a model is its ability to represent complex functions using a particular set of parameters. Ideally, we want our model to be identical to the function underlying the data, which is unknown. Thus, we try to control model complexity to ensure good generalization over unseen data, which is the performance measure of the model. Model complexity generally tends to increase with increase in training steps, due to increase in the set of values the parameters of the model can take. Regularization can be used to constrain model capacity for improved generalization. With regularization, we trade-off bias to reduce variance of the model. Some of the different ways to regularize a model are: L2 parameter norm penalties, L1 parameter norm penalties, dataset augmentation, dropout, early stopping, parameter sharing, etc.. In 3.5, we use 2 ways to control model complexity, viz. number of training steps, and L2 parameter norm penalties.

Here, we assume that examples with noisy labels are harder to classify than the correct ones which form some pattern or have a simpler data distribution. Thus, we hypothesize that overfitting leads to more noisy examples getting classified,

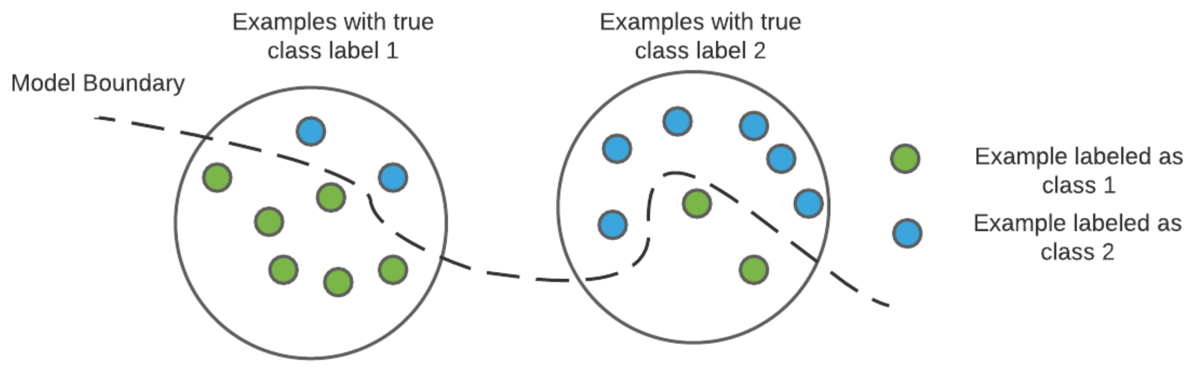

Examples with true class label 1
Examples with true class label 2
Model Boundary
Example labeled as class 1
Example labeled as class 2

Figure 2.1: Overfitting results in the model fitting examples with noisy labels.

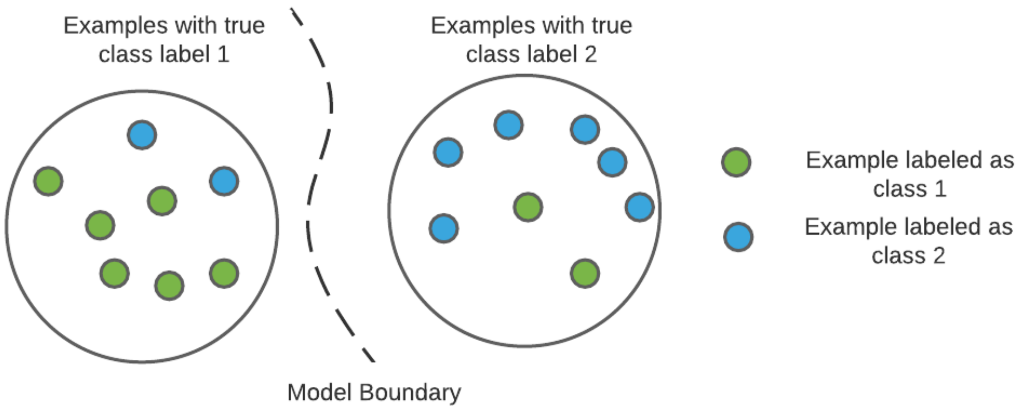

Examples with true class label 1
Examples with true class label 2
Example labeled as class 1
Example labeled as class 2
Model Boundary

Figure 2.2: Model with appropriate model complexity is able to generalize well without overfitting on noisy examples.

while a model with a controlled complexity will be robust to label noise. Figures 2.1 and 2.2 demonstrate this hypothesis.

2.1.2 Class Activation Maps

According to Zhou *et al.* (2016), class activation map for a particular class indicates the discriminative regions used by the CNN to identify that class. CAM is closely linked to the global average pooling(GAP) layer in CNNs, so we will give a brief overview of GAP first.

GAP layer in CNN is placed at the end of convolutional layers, which are usually followed by fully connected layers. It performs averaging over each of the feature map generated by the last convolutional layer. The class activation map

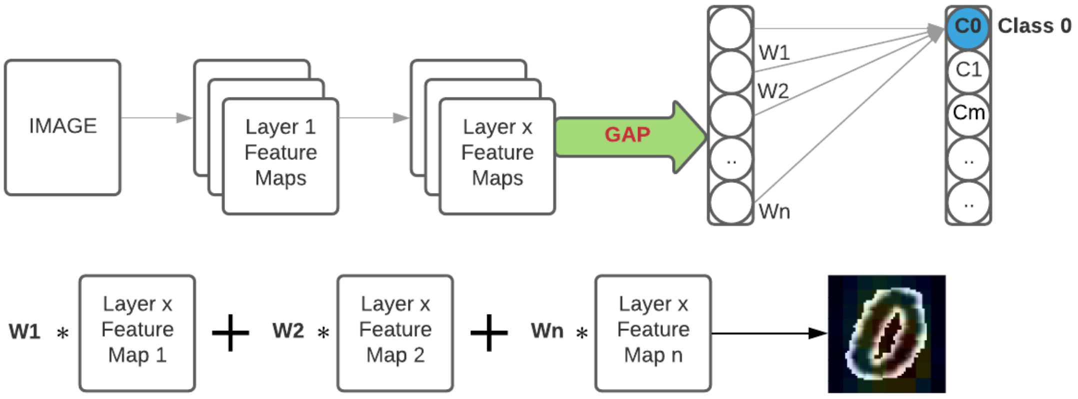


Figure 2.3: Class Activation Maps being visualized for Class 0 using the GAP layer in the CNN. The CAM for a class is a weighted average of last Conv layer feature maps using FC layer weights for the class. Figure adapted from Zhou *et al.* (2016).

for a particular class is the weighted sum of the feature maps generated by the last convolutional layer, where the weights are given by the weights of the fully connected layer connecting neurons to the particular neuron of that class. This is illustrated through 2.3

Let the last convolutional layer output be denoted by $f_k(x, y)$, where k is the filter index, and (x, y) are spatial co-ordinates. Let F_k denote the k^{th} element of the GAP layer, then $F_k = \sum_{x,y} f_k(x, y)$. Then, the pre-activations for class c S_c $= \sum_k F_k * w_{k,c} = \sum_k \sum_{x,y} f_k(x, y) * w_{k,c} = \sum_{x,y} \sum k f_k(x, y) * w_{k,c}$.

Defining M_c to be the Class activation map for class c, then $M_c(x, y) = \sum_k f_k(x, y) * w(k, c)$, thus, $S_c = \sum_{x,y} M_c(x, y)$.

2.2 Related Work

The problem of learning from noisy labels has been approached in different ways in the literature. The different approaches target different applications and make different underlying assumptions. There are some approaches which are theoretical in nature, where mathematical guarantees are given on noise tolerance under certain assumptions. Some approaches make assumptions on the nature of the noise present in the labels and model the label corruption in different ways. There are approaches which select clean instances from the noisy ones, or correct the noisy labels. Some approaches target applications for learning with web-data that has been assigned labels automatically, and thus are prone to be noisy. The approaches can be broadly classified as follows:

2.2.1 Theoretical Approaches

Approaches like Natarajan *et al.* (2013); Ghosh *et al.* (2017) are theoretical in nature, and make assumptions on the noise to prove noise-tolerance. Natarajan *et al.* (2013) assumes class-conditional label noise and shows efficient algorithms for ERM provided the loss satisfies some symmetry condition. It also proves existing methods like weighted logistic regression and biased-SVM to be noise-tolerant. Ghosh *et al.* (2017) shows that loss functions based on mean absolute value of error are inherently robust to label noise. One shortcoming of such approaches is that the assumptions these approaches make are highly restrictive. Due to the restricted validity of such approaches, they are not of high practical relevance.

2.2.2 Noise Modelling

Techniques like Sukhbaatar *et al.* (2014); Goldberger and Ben-Reuven (2016) introduce a linear layer that models the probability of corruption of the class label into another class. Hendrycks *et al.* (2018) assumes conditional independence of the clean label and noisy label given the example and uses some amount of clean data to mitigate the effect of noise. Patrini *et al.* (2017) introduces a loss correction factor which is dependent on corruption probabilities of a label to the other classes. This probability transition matrix is either already known or is estimated using a small clean dataset for validation. Patrini *et al.* (2017); Sukhbaatar *et al.* (2014); Goldberger and Ben-Reuven (2016) assume that the probability of corruption of an example is only dependent on the true class and the corrupted class, and independent of the individual training instance. These techniques assume that noisy labels are only dependent on the clean label and independent of individual instances given the clean label. These assumptions may not hold for large-scale web-data with autogenerated labels, for eg: for the Clothing1M dataset, as examined by Xiao *et al.* (2015). Such methods also have some difficulty in adapting to scenarios where the noisy labels are open-set Wang *et al.* (2018), i.e the examples do not belong to any of the class in the dataset. In comparison, our algorithms do not make any such assumption about the nature of the noise present in the dataset.

2.2.3 Instance Selection

Approaches like Jiang *et al.* (2017) use an extra teacher network which performs instance selection for the student network, which broadly takes the student network progress as input and returns weights for every example. Figure 2.5 shows the MentorNet architecture. Another approach Han *et al.* (2018) uses low loss as an indicator of a clean example and throws away high loss instances progressively. It uses 2 networks which exchange low loss training examples, which helps in reducing error flows. This technique essentially does hard (0-1) weighting of examples by throwing high loss instances away. Recent work Wang *et al.* (2018) introduced methods for tackling open-set noise, i.e. when the noisy examples cannot be relabeled correctly to some class in the dataset. They performed iterative learning using outlier detection, feature similarity matching and reweighting. Figure 2.4 depicts the core solution in Wang *et al.* (2018). Ren *et al.* (2018) uses meta-learning methods to learn the weights to be assigned to the examples. Our work does not depend on whether the noise is closed-set or open-set. We do soft weighting of examples and also give a principled way of iteratively reweighting examples, which is essentially a soft way of selecting clean instances while filtering out the noisy ones.

2.2.4 Weakly supervised Learning

Techniques like Guo *et al.* (2018); Xiao *et al.* (2015); Lee *et al.* (2018); Han *et al.* (2019) learn from automatically labeled web-data, which is an important application of learning from noisy labels. Guo *et al.* (2018) assumes that distribution density in the feature space is an indicator of the difficulty of the examples, and uses curriculum learning to obtain superior results on several web-scraped datasets. In one of our algorithms, we also implicitly train in the order of increasing difficulty

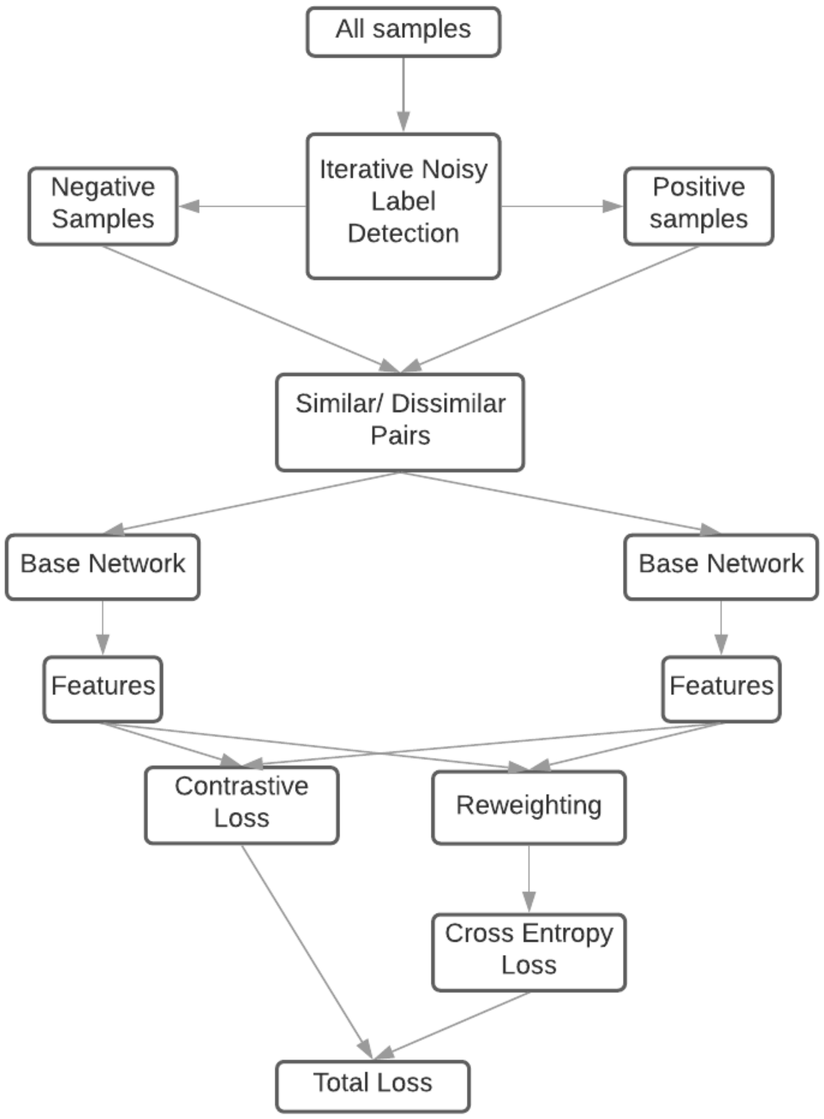


Figure 2.4: Architecture of the solution in Wang *et al.* (2018). Adapted from Wang *et al.* (2018).

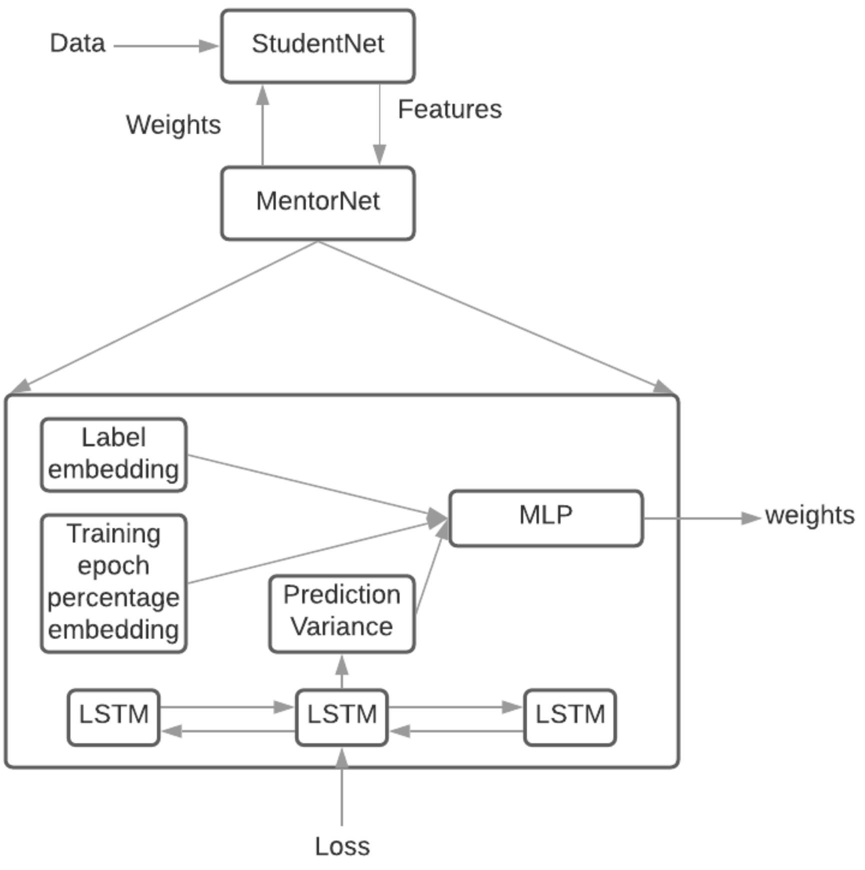


Figure 2.5: Architecture of the solution described in Jiang *et al.* (2017). Adapted from Jiang *et al.* (2017).

of examples, where we determine difficulty of an example by the complexity of the model required to classify it correctly. Lee *et al.* (2018); Han *et al.* (2019) iteratively refine or correct noisy labels by measuring similarity of an instance with representative instances of its class. One of the shortcomings of such methods is that they are not based on strong theoretical results, but are more ad-hoc in nature. This makes them hard to extend, compared to other methods.

2.2.5 Comparison

Our work deals with noisy labels as an instance selection problem, by filtering out examples with incorrect labels iteratively. We do not make any assumptions on the nature of noise, and do not limit our scope like the existing theoretical approaches. We also identify characteristics distinguishing clean examples from the noisy ones, and make novel observations regarding variation of model performance with complexity in the presence of noisy labels, both of which are independent reusable contributions. We make use of these novel insights in the DNN training process in our approach, which helps improve the performance of our iterative reweighting algorithm significantly. Thus, our work is significantly novel from prior work, and improves upon the performance of many algorithms in literature.

CHAPTER 3

Proposed Solution

In this work, we tackle the problem of learning from noisy labels by trying to select instances with correct labels and filtering out those with incorrect labels. Some of the advantages of this approach are that it does not make any assumptions about the noise present in the labels, and thus has wide applicability. Also, the approach taken here does not assume availability of clean data, making it more robust.

3.1 Problem Definition

In this work, we learn from labeled datasets which have incorrect labels to some extent in the training data. One of the challenges in this problem is that there is no equivalent of clean validation data, and thus it is difficult to measure how well a model generalizes to unseen data. The metric that we can measure during training: training accuracy does not represent the actual accuracy, due to label corruption. This necessitates development of alternate metrics to measure while training, in the absence of clean validation data. Further, there are no theoretically grounded approaches that work in diverse conditions, thus solutions that we develop are backed by empirical evidence, but not by rigorous mathematical guarantees.

3.2 Characteristics differentiating examples with clean and noisy labels

3.2.1 Predicted probabilities

We hypothesize that probability of an example for its class as predicted by the model could be used as a characteristic to separate between clean and noisy labels. This hypothesis is motivated by Arpit *et al.* (2017). This paper states that DNNs, which are capable of fitting even random noise, fit simpler patterns present in data in the initial stages of training.

From this observation, we deduced that in the initial stages of training, the DNN will fit the inherent patterns present in the data, which correspond to the clean labels, assuming that the examples with clean labels are in majority. This led us to hypothesize that probability values predicted for clean examples would be higher than those predicted for noisy examples. This was verified through experiments on multiple datasets: MNIST, CIFAR-10, CIFAR-100.

We make one more important observation from the experiments, that the separation between probability value distributions corresponding to examples with clean and noisy labels increases with reduction in noise levels.

3.2.2 Class Activation Maps

We use the Global average pooling(GAP) layer in the network to build class activation maps, which are essentially weighted sums of feature maps used as input for the GAP layer. Class activation maps act as some form of attention over the prominent objects in an image. Here, we use class activation maps to visualize the differences between examples with clean and noisy labels. We also try to get a more intuitive understanding of how attention on images changes with label noise.

We use CAMs to visualize the attention of the CNN model on images. We hypothesize that the CAMs for examples with the correct labels should focus on the important regions in the image, which is required in order to classify the image correctly into that class. On the other hand, for examples with wrong labels, we expect that the attention would be random and would not focus on any particularly important region in the image. We confirm these hypotheses via experiments and visualizations(Figure: 3.1).

3.3 Regularization

Model regularization helps limit the complexity of a model by restricting its parameters. This means that a model trained with high regularization will be able

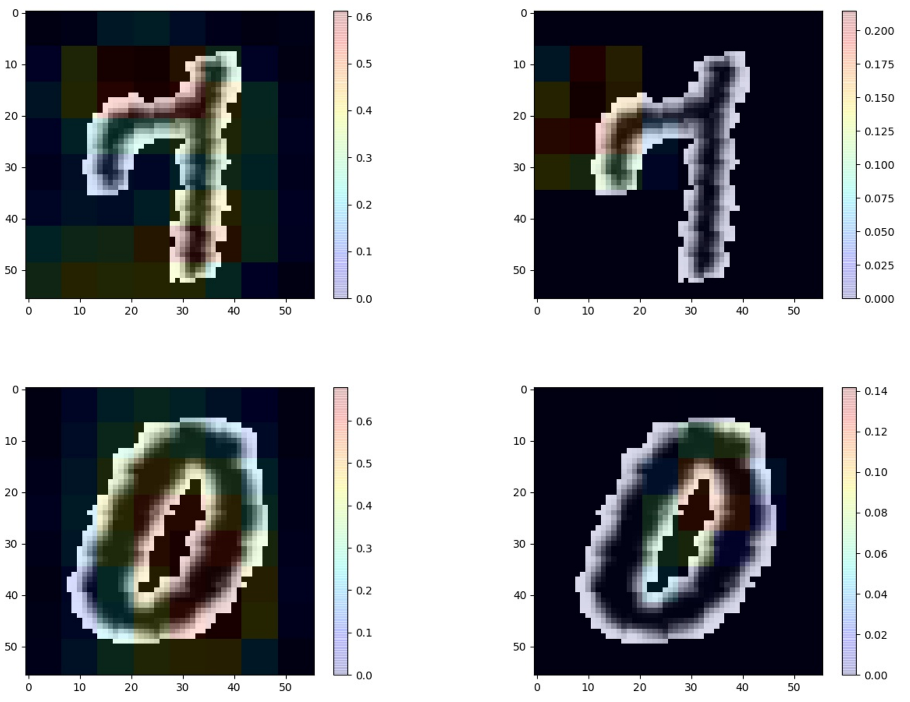


Figure 3.1: Class Activation Maps overlaid onto images from the MNIST dataset. In each row, the first heatmap depicts the CAM for the correct class and the second one depicts the CAM for an incorrect class indicated by the noisy label.

to only fit simpler patterns than those that can be fit by a model with low/no regularization. In this problem setting, majority of examples with clean labels correspond to simpler patterns in data, whereas examples with noisy labels correspond to harder patterns in data, along with clean examples which are hard to classify.

Figure 3.2 shows how regularization enables some amount of separation of clean examples from noisy ones. At higher values of regularization, the number of clean examples classified is higher than noisy ones, because the model can only fit simpler patterns in data. As regularization decreases in the training process, the model starts to fit noisy data. Thus, the fraction of clean examples classified by the model out of the total classified decreases with decreasing regularization. Algorithm 3.6.2 is directly based on this observation.

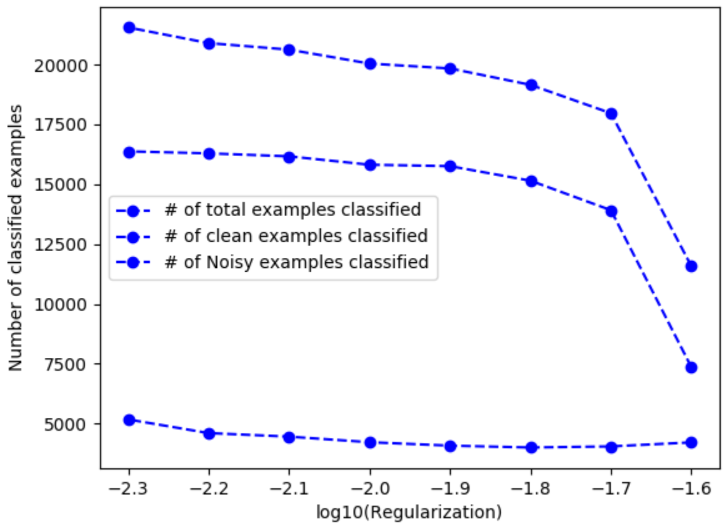


Figure 3.2: Plot of number of classified examples by the model on CIFAR-10 dataset with 45% pair-flip noise: total number of correctly classified examples, examples with clean labels, examples with noisy labels.

3.4 Reweighting

Training on datasets with noisy labels can achieve good performance if examples with noisy labels are removed from the training set. Section 3.2 states that clean examples can be characterized as having higher class prediction probability from the DNN model than examples with noisy labels. This characterization motivated the weighting of examples based on their prediction probability values obtained from the model. So, we use a weighted loss function for training:

$$L = \sum_{i=1}^{N} l_i w_i$$

where l_i and w_i correspond to loss and weight for the i^{th} training example and N is size of the training dataset. We tried different functions like linear, quadratic, exponential functions 3.3 for mapping prediction probability into the weight for an example.

Based on the above arguments, it is possible to formulate a training algorithm based on reweighting, independent of the DNN model used. There are some shortcomings associated with this reweighting approach that we observed. First, the predicted probability by the model for an example which is used to determine

$$calc\_weight(p_{example}, p_{max}, p_{min}) = \frac{p_{example} - p_{min}}{p_{max} - p_{min}}$$
$$calc\_weight(p_{example}, p_{max}, p_{min}) = (\frac{p_{example} - p_{min}}{p_{max} - p_{min}})^2$$
$$calc\_weight(p_{example}, p_{max}, p_{min}) = \frac{exp(\frac{p_{example} - p_{min}}{p_{max} - p_{min}}) - 1}{e - 1}$$

Figure 3.3: Linear, Quadratic and Exponential example weighting functions. $p_{example}$ is the probability value for the example, p_{max} and p_{min} are maximum and minimum probability values across the entire training dataset.

the weight of an example should be reasonable, i.e. lower for a noisy and higher for a clean labeled example. Initially, at the start of training, the model predicts randomly and thus the estimates are bad. According to Arpit *et al.* (2017), initially the model will learn the simpler patterns in data, thus improving the probability estimate. But, the model will eventually overfit on the noisy data with large number of training steps deteriorating the estimates.

This implies that there is a training stage in between the start and overfitting stage, where the probability estimates are reasonable, and the model has fit simple patterns in data, without overfitting on noisy instances. Thus, this stage is a good one to start our reweighting algorithm. The problem is that this stage depends on difficulty of the dataset, learning rate etc. and thus varies across different datasets and identifying it in every dataset without access to a clean validation set is tough. Section 3.5 tries to present a solution to this problem, using insights from the DNN training process.

3.5 Analyzing model performance with varying model complexity

In this section, we present novel observations regarding the training of DNNs. We analyze model performance with varying model complexity, and observe a common pattern emerging irrespective of datasets and different noise levels. Further, we observe the same pattern independently using 2 ways to control model complexity, viz. the number of training epochs and L2 regularization.

In the absence of noise in labels, the difference in model accuracy decreases with increasing model complexity. In the presence of noise in labels, the difference in model accuracy decreases upto a certain point, then increases in some range and then decays to zero with increasing model complexity.

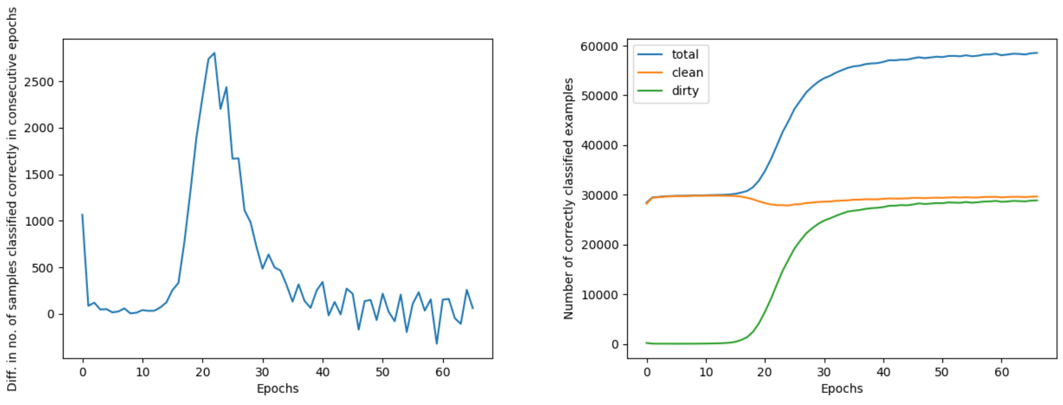
We try to explain this by reasoning about the DNN training process. Initially the rate of change of accuracy with model complexity is high, as the decision boundary of the model is far from optimal and even a small change in model complexity can result in a higher number of examples getting correctly classified. The decrease in the rate of change of accuracy is similar to diminishing returns as increase in accuracy on efforts to change the decision boundary. The increase in the rate of change of accuracy, which does not happen in the case of clean labels, can be explained as the model trying to fit examples with noisy labels, which causes the increase in the rate.

The novel observations that we make are important in the reweighting algorithms that we introduce in the next section.

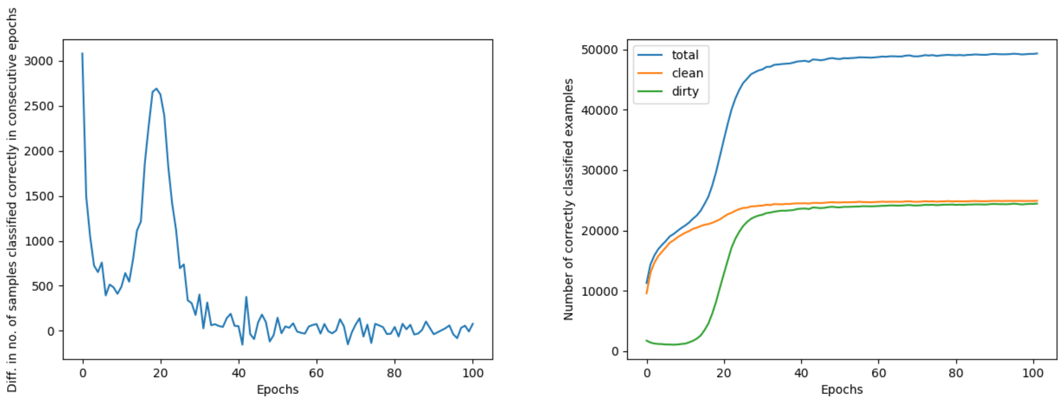
3.5.1 Using number of epochs to control model complexity

In this method, change in model accuracy is tracked with the number of training epochs. With increasing in training epochs, the model complexity increases, and the change in model accuracy follows a pattern as described above.

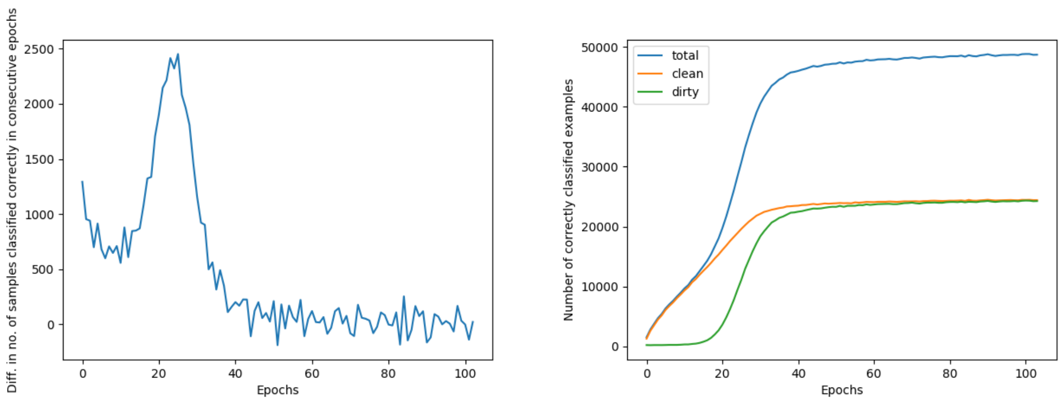
The graphs in 3.4 show the difference in accuracies with epochs as well the number of examples classified correctly by the model according to correct, incorrect labels and overall as well. From the figures, it can be seen that initially, majority of the examples getting classified correctly according to the labels have correct labels, and thus we can infer that the model learns the inherent pattern in the data. When the difference in accuracies starts increasing, the model starts overfitting on examples with noisy labels, which can be seen in the figure. Thus, the figures provide evidence for the observation and support the explanation above.



(a) MNIST- 50% symmetric noise



(b) CIFAR 10- 50% symmetric noise



(c) CIFAR 100- 50% symmetric noise

Figure 3.4: In each row, the first figure shows the difference in number of training examples classified between 2 consecutive epochs and the second figure shows the % of clean and noisy examples in all training examples classified correctly according to the noisy training set.

3.5.2 Using regularization to control model complexity

In this method, models are trained to saturation with a certain regularization, which limits the complexity of the model. This is implemented efficiently by training a single model, by successively training it to saturation on a particular regularization value, and changing the regularization value.

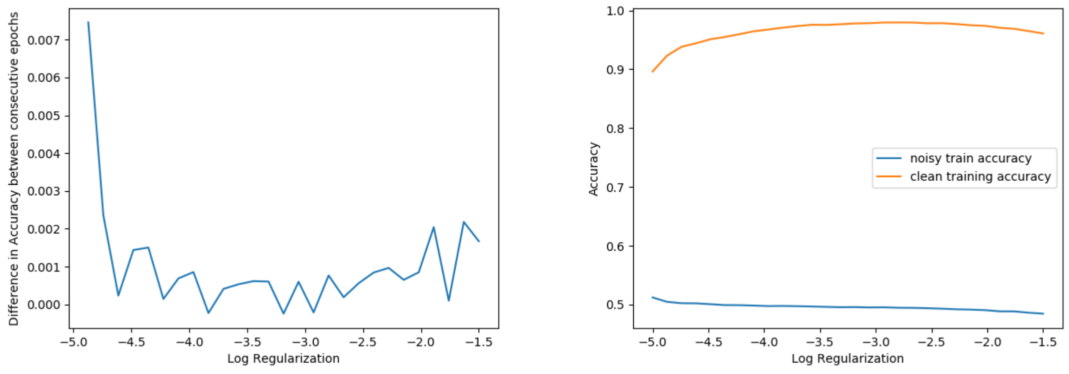
The figures in 3.5 show the difference in accuracies with models trained with different regularization values and accuracies of the models calculated using noisy labels and clean labels(clean labels are used only for analysis here). The model complexity increases from right to left, due to lower regularization values. Interpreting the curves from right to left, the graph depicting difference in accuracy follows the same pattern as 3.4. The other shows that the accuracy as calculated with both noisy labels as well as clean labels increases initially, indicating learning of the correct underlying patterns in data. The accuracy calculated on clean labels decreases, as the difference in accuracy increases, indicating that the model starts fitting data with noisy labels. In this way, 3.5 also substantiates the explanation for the observation.

3.6 Algorithms

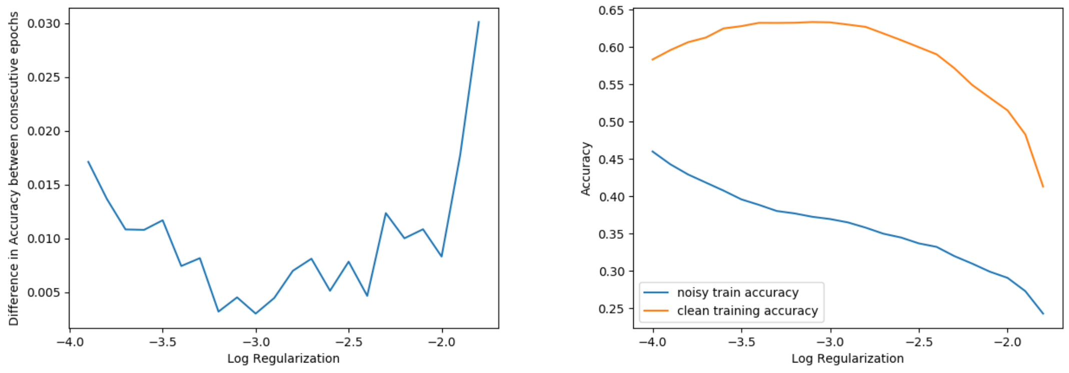
In this section, we describe the pseudocode for 2 algorithms that we have developed. These algorithms use all of the ideas that we have described in detail in this chapter. The 2 algorithms are based on reweighting and regularization.

3.6.1 Reweighting Algorithm

Here, we present the systematic iterative reweighting algorithm which is a training algorithm to learn in the presence of noisy labels. The main idea behind this algorithm is that, starting from reasonable probability estimates for examples, we can alternately train and weigh loss as a function of probability estimates, which will further improve the estimates. One limitation of this algorithm is that it tends to filter out hard examples having correct labels along with examples having incorrect labels. We present a flow diagram of the algorithm 3.6 followed



(a) MNIST- 50% symmetric noise



(b) CIFAR 10- 50% symmetric noise

Figure 3.5: In each row, the first figure shows the difference in number of training examples classified correctly between models trained to saturation with 2 consecutive regularization values and the second figure shows the accuracy with models trained with different values of regularization according. The 2 curves shows accuracies calculated using noisy labels, and the other curve is calculated using clean labels.

by the actual algorithm 3.6.1.

```
Input: Noisy data: D = $\{x_i, y_i\}_{i=1}^{i=N}$, learning Rate $\eta$, Number of epochs
       $E_{total}$ Optimal epoch to start reweighting $E_{opt}$
Output: Model parameters $\theta$
1 $\forall x \in \{1, \cdots, N\}, W_x \leftarrow 0.5$
2 for $i \leftarrow 1$ to $E_{total}$ do
3     Train model with parameters $\theta$, learning rate $\eta$, for 1 epoch using the
       weighted loss $L = \sum_{j=1}^{N} l_j W_j$ using weights W, where $W_j$ and $l_j$ are
       the weight and cross-entropy loss for $j^{th}$ example
4     if $i \geq E_{opt}$ then
5         $\forall j \in \{1, \cdots, N\}$ $p_j = Pr(\hat{y}_j = k | x_j, \theta)$, where $x_j$ belongs to class $k$
           and $\hat{y}$ denotes the model's prediction
6         $p_{max}, p_{min} \leftarrow max_j(p_j), min_j(p_j)$, where $j \in \{1, \cdots, N\}$
7         $\forall j \in \{1, \cdots, N\}, W_j \leftarrow calc\_weight(p_j, p_{max}, p_{min})$ //
           calc_weight calculates the weight for an example using one of the
           functions in 3.3.
8     end
9 end
```

Algorithm 1: Systematic Iterative Reweighting Algorithm.

3.6.2 Regularization based algorithm

Here we present a training algorithm based on regularization, which assigns weight to examples, with the objective of filtering out examples with noisy labels. The basic idea underlying this algorithm is that model complexity can be controlled by regularization, and models of different complexities can classify examples easier to

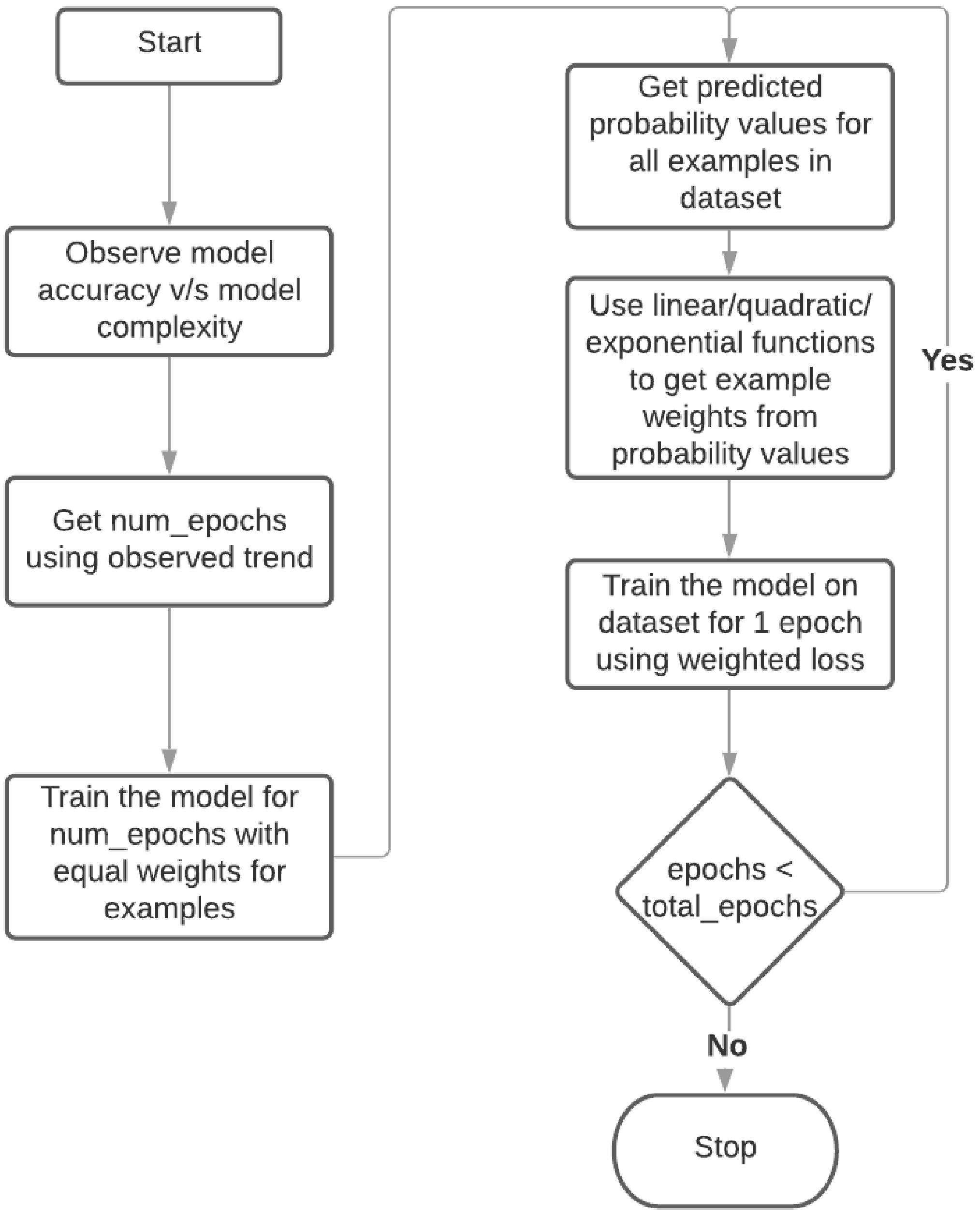

Start
Observe model accuracy v/s model complexity
Get num_epochs using observed trend
Train the model for num_epochs with equal weights for examples
Get predicted probability values for all examples in dataset
Use linear/quadratic/ exponential functions to get example weights from probability values
Train the model on dataset for 1 epoch using weighted loss
epochs < total_epochs
Yes
No
Stop

Figure 3.6: Flowchart for the algorithm 3.6.1.

classify than a particular difficulty.

```
   Input: Noisy data: D = $\{x_i, y_i\}_{i=1}^{i=N}$
1  Hyperparameters: Regularization values V in descending order, such
     that $|V| = k$, Epochs = E
   Output: Example weights W, a vector of length N, with $0 \leq W_i \leq 1 \; \forall i$
             $1 \leq i \leq N$
2  Initialize DNN model parameters $\theta$
3  $T \leftarrow 0$ //Denotes number of examples which are given non-zero weight
4  $\forall x \in \{1, \cdots, N\}, W_x \leftarrow 0$
5  for $i \leftarrow 1$ to $k$ do
6      $\lambda \leftarrow V[i]$
7      for $e \leftarrow 1$ to $k$ do
8          Train the model with parameters $\theta$ for 1 epoch using
             Cross-Entropy loss and L2 regularization with parameter $\lambda$.
9          $X_{curr} \leftarrow$ Set of examples classified correctly at end of epoch e
10         $X_{prev} \leftarrow$ Set of examples having non-zero weight at end of epoch
             $e - 1$
11         $X_{new} \leftarrow X_{curr} \setminus X_{prev}$
12         w $\leftarrow$ 1 - $\frac{T}{N}$
13         $\forall x \in X_{new}, W_x \leftarrow w$
14         $T \leftarrow T + |X_{new}|$
15     end
16 end
```

Algorithm 2: Weighting examples using Regularization.

3.7 Generalization across noise models

Existing approaches like Han *et al.* (2018) and Yu *et al.* (2019) show results on clean datasets by injecting controlled noise in them. We evaluate these methods on real-world noisy datasets which have unknown noise distribution, in order to see how performance on controlled noise extends to a real-world dataset, with a possibly complex underlying noise model. From the results in 4.4, we can see that the methods do not perform well on a real-world noisy dataset, thus showing that a method performing well on controlled noise may not necessarily perform equally well on real-world noisy dataset.

CHAPTER 4

Performance Study

In this chapter, we describe our experimental setup and experiments. We also analyze the performance of our algorithms when subject to a variety of experiments. Then, we compare those results with methods in the literature and analyze the advantages and disadvantages of our method.

4.1 Experiments

In this section, we evaluate robustness of the iterative reweighting algorithm 3.6.1 on multiple datasets, present experimental details, results and ablation studies for our model.

4.1.1 Datasets

The datasets used for evaluation of the model are: MNIST, CIFAR-10, CIFAR-100. Noisy datasets are constructed from MNIST, CIFAR-10 and CIFAR-100 datasets by removing some randomly picked training samples from a class and putting them either all into 1 other class or by randomly assigning a label to them which is different from the original class. We perform experiments using 3 specific noise cases for MNIST, CIFAR-10 and CIFAR-100: 50% symmetric noise, 20% symmetric noise and 45% pair-flip noise.

4.1.2 Implementation Details

We implement all methods and algorithms using PyTorch Paszke *et al.* (2019) and conduct experiments on a NVIDIA GeForce GTX 1080 GPU. Pytorch is chosen as the software package for implementation of our algorithms due to ease of programming owing to its imperative nature, as compared to declarative frameworks like

Tensorflow Abadi *et al.* (2016), which is quite hard to use. For all experiments, Adam optimizer is used with the default learning rate(0.001), with a batch size of 128 and we run for 200 epochs for each experiment. We use the neural network architecture as described in Han *et al.* (2018) for our experiments. We use data augmentation in our experiments, and use the horizontal flip and random rotation transformation with a maximum rotation of 30 degrees. The epoch at which reweighting is started is determined from the plots 3.4

We test our algorithm on the task of multi-class classification, and thus we use cross-entropy as the loss function to track training progress. We use the accuracy on a clean test dataset as our performance metric. We also observe weight histograms for clean and noisy labeled examples, which allows us to analyze the algorithm's performance.

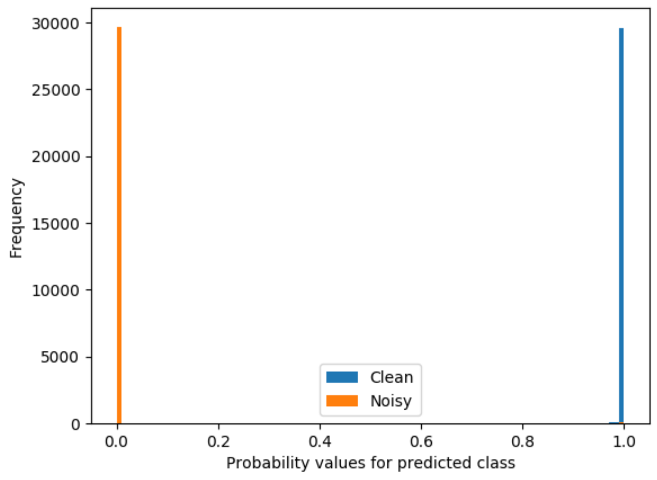
4.2 Results and Analysis

We will discuss the results obtained by running the algorithm 3.6.1 on the datasets MNIST, CIFAR-10, CIFAR-100 on 3 noise cases: 45% pair-flip noise, 50% symmetric noise, 20% noise. Tables 4.1, 4.2, 4.3 shows the accuracy obtained by the model on the clean test set of the individual datasets. Figure 4.2 shows the accuracy curves which include the training accuracy calculated with noisy labels, training accuracy containing clean labels and the accuracy on the clean test set for the 50% symmetric case. Figure 4.1 shows the histograms of probability values predicted by the model at the end of the 200 epoch training process. We compare the performance of our method with Patrini *et al.* (2017); Jiang *et al.* (2017); Han *et al.* (2018) as the baselines.

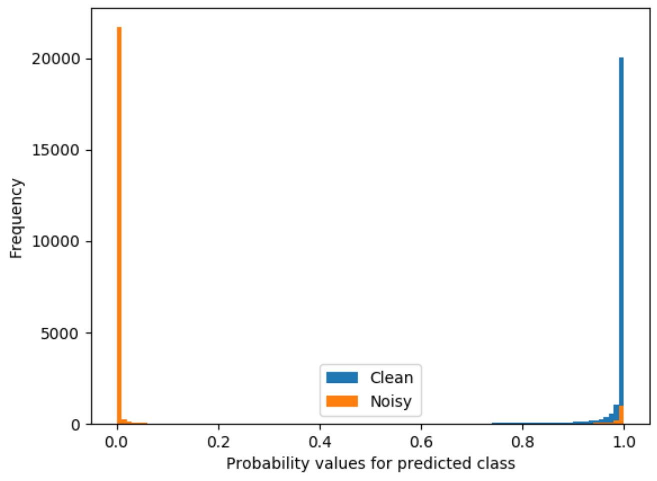
Method\ Noise	Pair-45	Symmetric-50	Symmetric-20
F-Correction Patrini *et al.* (2017)	0.24	79.61	98.80
Mentornet Jiang *et al.* (2017)	80.88	90.05	96.70
Co-teaching Han *et al.* (2018)	87.63	91.32	97.25
Ours (3.6.1)	**98.76**	**98.9**	**99.42**

Table 4.1: Comparison of Results on MNIST test data after 200 epochs.

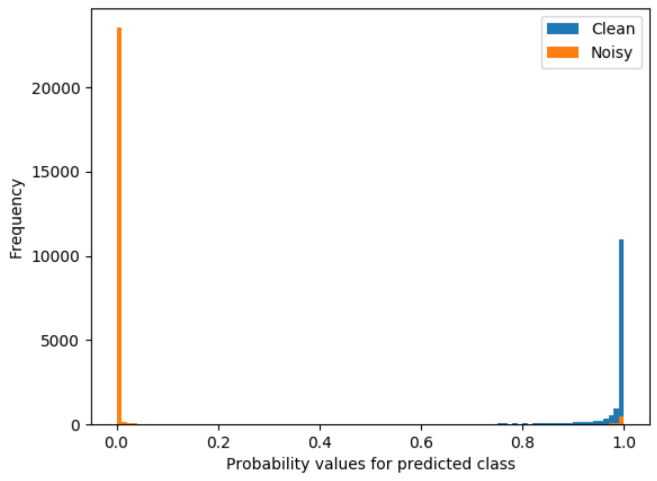
Results on MNIST: The reweighting algorithm applied on the model is able to separate examples with clean labels from those with noisy labels successfully.



(a) MNIST Dataset

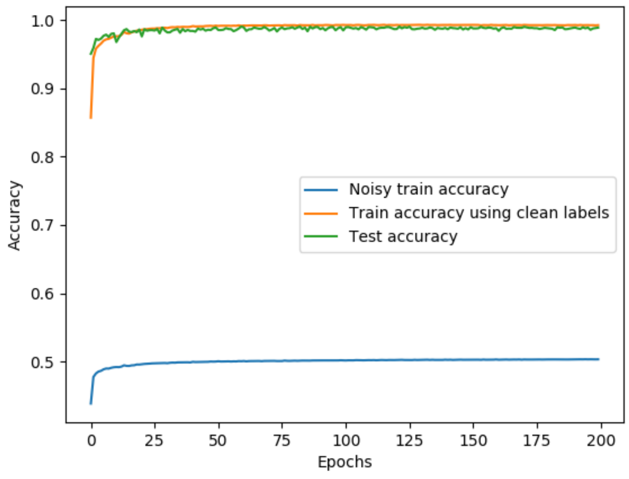


(b) CIFAR-10 Dataset

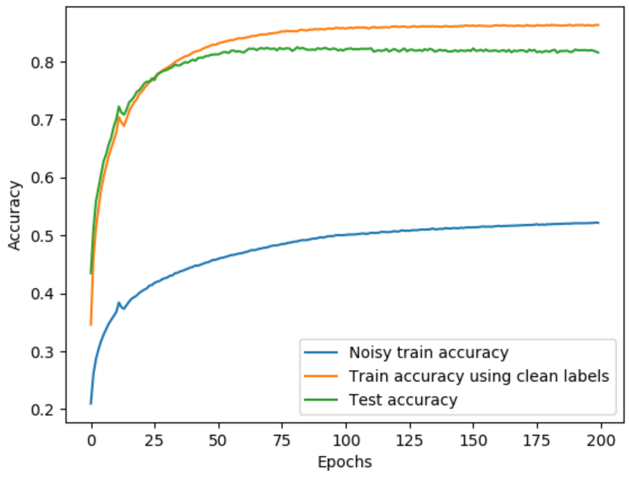


(c) CIFAR-100 Dataset

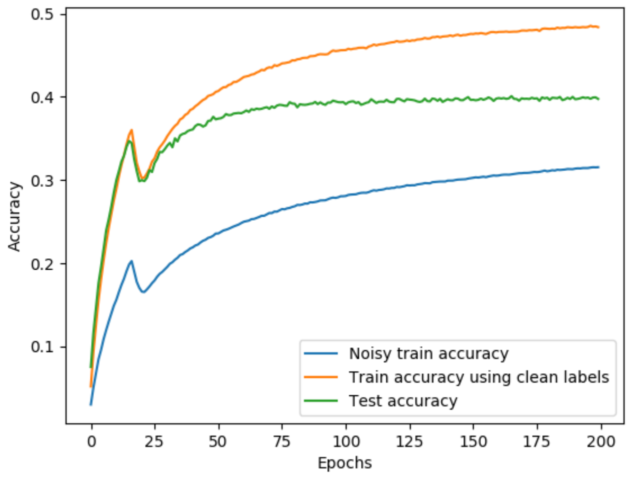
Figure 4.1: Probability histograms after the training for 200 epochs for 3 different dataset with 50% symmetric noise.



(a) MNIST Dataset



(b) CIFAR-10 Dataset



(c) CIFAR-100 Dataset

Figure 4.2: Accuracy curves representing training accuracy calculated with noisy labels, clean labels and test accuracy for 200 epochs for 3 datasets with 50% symmetric noise.

Method\ Noise	Pair-45	Symmetric-50	Symmetric-20
F-Correction Patrini *et al.* (2017)	6.61	59.83	84.55
Mentornet Jiang *et al.* (2017)	58.61	71.10	80.76
Co-teaching Han *et al.* (2018)	72.62	74.02	82.32
Ours (3.6.1)	**81.77**	**81.57**	**87.7**

Table 4.2: Comparison of Results on CIFAR-10 test data after 200 epochs.

Method\ Noise	Pair-45	Symmetric-50	Symmetric-20
F-Correction Patrini *et al.* (2017)	1.60	41.04	61.87
Mentornet Jiang *et al.* (2017)	31.60	39.00	52.13
Co-teaching Han *et al.* (2018)	34.81	41.37	54.23
Ours (3.6.1)	**40.00**	39.77	58.07

Table 4.3: Comparison of Results on CIFAR-100 test data after 200 epochs.

This is clear from the probability histograms, which show clean separation of clean and noisy examples. This is also reflected in the high accuracy of $\approx 99\%$ that the model achieves irrespective of the noise case considered. For high noise cases, the accuracy obtained by our algorithm is $\approx 10\%$ higher than the baseline, demonstrating effectiveness of the method to separate noisy examples.

Results on CIFAR-10: Algorithm 3.6.1 achieves 5%-10% higher accuracies than the baseline with a corresponding good separation of examples, which is visible through the probability histograms.

Results on CIFAR-100: Algorithm 3.6.1 performs better than the baseline in 1 noise case and achieves comparable performance on the other noise cases. The major difficulty in training on CIFAR-100 is the low number of clean training examples available per class. We think that such a problem would not occur in real-world noisy datasets, as they mostly tend to be large in size, due to low cost required to collect them, for eg: images obtained from search engines using queries as labels.

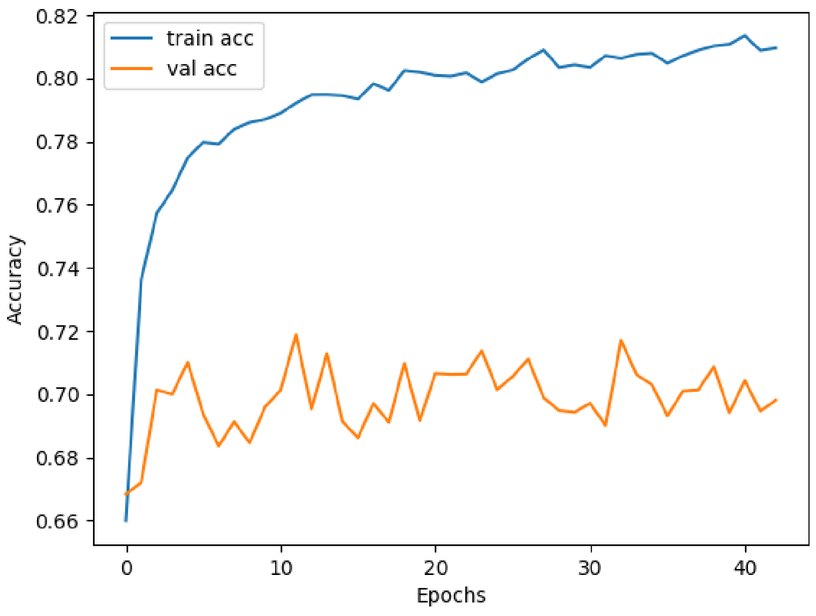
4.2.1 Experiments on Real-world noisy data

We experiment on real-world noisy dataset Clothing-1M introduced by Xiao *et al.* (2015). In the dataset, images are scraped from the web, and tags/ queries are used to assign labels from predefined categories. Preliminary experiments using the reweighting algorithm does not show any significant improvements over the

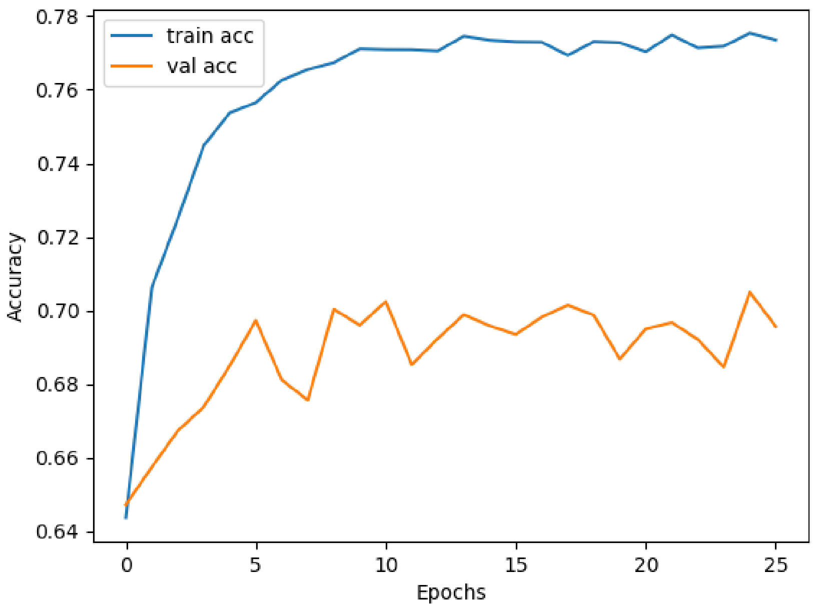
Dataset\ Algorithm	Cross-Entropy	Han *et al.* (2018)	Yu *et al.* (2019)	Ours
Clothing-1M Xiao *et al.* (2015)	69.54%	68.11	41.32	70.11

Table 4.4: Results of Han *et al.* (2018); Yu *et al.* (2019) and 3.6.1 on the Clothing-1M dataset, which is a real-world noisy dataset.

baseline performance, as can be seen from 4.3.



(a) Performance using cross-entropy loss on Clothing-1M



(b) Performance using algorithm 3.6.1 on Clothing-1M

Figure 4.3: Performance comparison on Clothing-1M dataset Xiao *et al.* (2015) between 3.6.1 using cross-entropy loss trained model as baseline.

4.2.2 Ablation Study

Observation of DNN Training on Clean Dataset:
In Section 3.5, we observed the DNN training process on dataset with noisy labels and derived insights regarding the training process. We demonstrate here that running on a clean dataset gives a pattern 4.4 different from that obtained in 3.4, proving that the pattern obtained is a characteristic of noise in the labels.

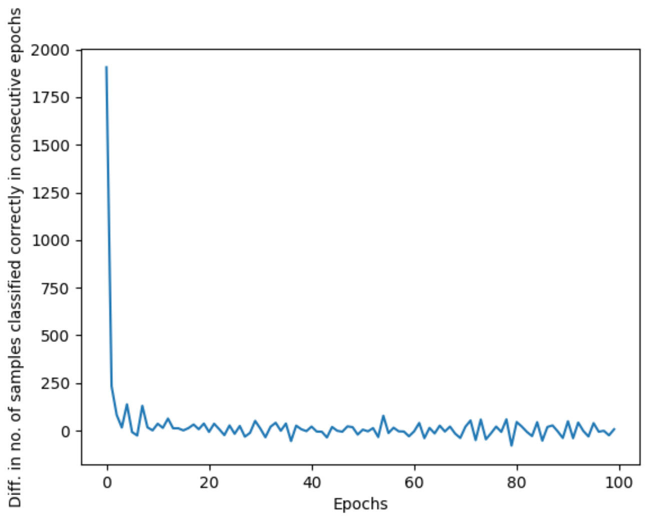
4.3 Discussion

We start by finding distinguishing factors between examples having correct and incorrect labels, and find 2 factors, first, in the form of high predicted probability examples, and another based on attention over the image, based on CAMs.

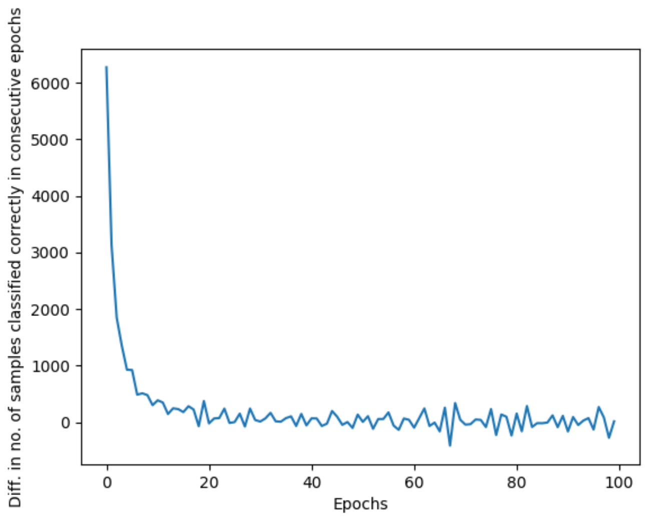
We make novel observations reagrding variation of model performance with model complexity for DNN training in the presence of noisy labels. These observations about the training process are obtained using 2 measures of model complexity, viz. number of training epochs and regularization. These observations and insights about the DNN training process help in improving the performance of our algorithms, but can be used independently as well.

Our algorithm 3.6.1 performs better than the baselines Patrini *et al.* (2017); Jiang *et al.* (2017); Han *et al.* (2018) in every dataset and noise case, except 2 noise cases in CIFAR-100 dataset, where it gives a performance comparable with the baselines. The baseline Patrini *et al.* (2017) devises approach making assumptions on the nature of the noise, which is not the case with our algorithm. The baselines Han *et al.* (2018); Jiang *et al.* (2017) drops instances based on the low-loss criterion, which is equivalent to giving hard weights of 0 and 1 to examples. Also, number of instances to be dropped follows a fixed schedule. In contrast, our method does explicit soft reweighting of examples, without following any fixed schedule. Also, starting of the reweighting is done using a systematic way based on concrete experimental evidence about the DNN training process.

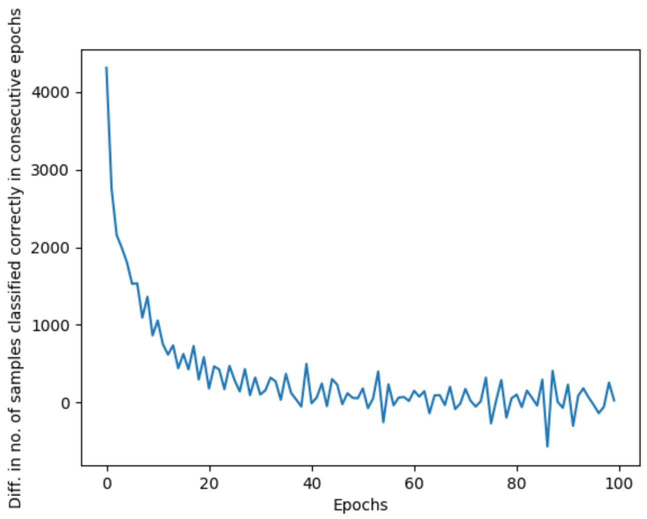
We perform experiments to show that methods like Han *et al.* (2018); Yu *et al.* (2019) which work well on controlled noise do not work well on real-world noisy



(a) MNIST Dataset



(b) CIFAR-10 Dataset



(c) CIFAR-100 Dataset

Figure 4.4: Difference between number of examples classified by the model in consecutive epochs on clean datasets.

data, with some even performing significantly worse than the standard baseline using only cross-entropy loss 4.4. Preliminary experiments on real-world noisy data suggest that our algorithms give some improvement, but do not significantly improve over the baseline method.

CHAPTER 5

Conclusion and Future Work

In this work, we make multiple contributions to the area of learning from noisy labels. We define 2 characteristics, viz. class-prediction probability and the quality of CAM visualizations, both of which distinguish examples with correct labels from those with incorrect labels. One of the future directions from this work is the problem of using characteristics which are qualitative in nature, for eg: the quality of CAMs for correct and incorrectly labeled instance, to devise an algorithm to filter noisy instances.

We take the approach of selecting clean instances from the noisy ones, and develop 2 novel training algorithms based on iterative reweighting and regularization. The iterative reweighting algorithm gives superior performance than many of the recent approaches in this area.

Also, we make novel observations regarding variations in DNN performance with varying DNN complexity, which can be controlled by regularization or by the number of training steps. These observations provide us novel insights into the DNN training process. These observations provide a systematic way to boost the performance of the iterative reweighting algorithm. Although we have used this observation with the reweighting algorithm, this observation is completely independent of the algorithm and can be used independently in different contexts.

One of our results shows that some of the existing methods that work quite well in controlled label noise settings fail with real-world label noise. Preliminary experiments indicate that our method also does not generalize well to real-world label noise. This points us to a bigger research question regarding generalization of models across different types of noise in data, which is quite important, given the varying nature of noisy labels in real-world data.

REFERENCES

1. **Abadi, M.**, **P. Barham**, **J. Chen**, **Z. Chen**, **A. Davis**, **J. Dean**, **M. Devin**, **S. Ghemawat**, **G. Irving**, **M. Isard**, *et al.*, Tensorflow: A system for large-scale machine learning. *In 12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*. 2016.

2. **Arpit, D.**, **S. Jastrzębski**, **N. Ballas**, **D. Krueger**, **E. Bengio**, **M. S. Kanwal**, **T. Maharaj**, **A. Fischer**, **A. Courville**, **Y. Bengio**, *et al.*, A closer look at memorization in deep networks. *In Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017.

3. **Frénay, B.** and **M. Verleysen** (2013). Classification in the presence of label noise: a survey. *IEEE transactions on neural networks and learning systems*, **25**(5), 845–869.

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

5. **Goldberger, J.** and **E. Ben-Reuven** (2016). Training deep neural-networks using a noise adaptation layer.

6. **Guo, S.**, **W. Huang**, **H. Zhang**, **C. Zhuang**, **D. Dong**, **M. R. Scott**, and **D. Huang**, Curriculumnet: Weakly supervised learning from large-scale web images. *In Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.

7. **Han, B.**, **Q. Yao**, **X. Yu**, **G. Niu**, **M. Xu**, **W. Hu**, **I. Tsang**, and **M. Sugiyama**, Co-teaching: Robust training of deep neural networks with extremely noisy labels. *In Advances in neural information processing systems*. 2018.

8. **Han, J.**, **P. Luo**, and **X. Wang**, Deep self-learning from noisy labels. *In Proceedings of the IEEE International Conference on Computer Vision*. 2019.

9. **Hendrycks, D.**, **M. Mazeika**, **D. Wilson**, and **K. Gimpel**, Using trusted data to train deep networks on labels corrupted by severe noise. *In Advances in neural information processing systems*. 2018.

10. **Jiang, L.**, **Z. Zhou**, **T. Leung**, **L.-J. Li**, and **L. Fei-Fei** (2017). Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. *arXiv preprint arXiv:1712.05055*.

11. **Lee, K.-H.**, **X. He**, **L. Zhang**, and **L. Yang**, Cleannet: Transfer learning for scalable image classifier training with label noise. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018.

12. **Natarajan, N.**, **I. S. Dhillon**, **P. K. Ravikumar**, and **A. Tewari**, Learning with noisy labels. *In Advances in neural information processing systems*. 2013.

13. **Paszke, A.**, **S. Gross**, **F. Massa**, **A. Lerer**, **J. Bradbury**, **G. Chanan**, **T. Killeen**, **Z. Lin**, **N. Gimelshein**, **L. Antiga**, *et al.*, Pytorch: An imperative style, high-performance deep learning library. *In Advances in Neural Information Processing Systems*. 2019.

14. **Patrini, G.**, **A. Rozza**, **A. Krishna Menon**, **R. Nock**, and **L. Qu**, Making deep neural networks robust to label noise: A loss correction approach. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017.

15. **Ren, M.**, **W. Zeng**, **B. Yang**, and **R. Urtasun** (2018). Learning to reweight examples for robust deep learning. *arXiv preprint arXiv:1803.09050*.

16. **Sukhbaatar, S.**, **J. Bruna**, **M. Paluri**, **L. Bourdev**, and **R. Fergus** (2014). Training convolutional networks with noisy labels. *arXiv preprint arXiv:1406.2080*.

17. **Thomee, B.**, **D. A. Shamma**, **G. Friedland**, **B. Elizalde**, **K. Ni**, **D. Poland**, **D. Borth**, and **L.-J. Li** (2016). Yfcc100m: The new data in multimedia research. *Communications of the ACM*, **59**(2), 64–73.

18. **Wang, Y.**, **W. Liu**, **X. Ma**, **J. Bailey**, **H. Zha**, **L. Song**, and **S.-T. Xia**, Iterative learning with open-set noisy labels. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018.

19. **Xiao, T.**, **T. Xia**, **Y. Yang**, **C. Huang**, and **X. Wang**, Learning from massive noisy labeled data for image classification. *In Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.

20. **Yu, X.**, **B. Han**, **J. Yao**, **G. Niu**, **I. W. Tsang**, and **M. Sugiyama** (2019). How does disagreement help generalization against label corruption? *arXiv preprint arXiv:1901.04215*.

21. **Zhang, C.**, **S. Bengio**, **M. Hardt**, **B. Recht**, and **O. Vinyals** (2016). Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*.

22. **Zhou, B.**, **A. Khosla**, **L. A.**, **A. Oliva**, and **A. Torralba** (2016). Learning Deep Features for Discriminative Localization. *CVPR*.