Learning with Long-Tailed Noisy Labels

A dissertation submitted in
partial fulfilment for the degree of

Master of Technology

in

Computer Science

by

Sarbajit Dey
Roll no. - [**CS2226**]

under the supervision of

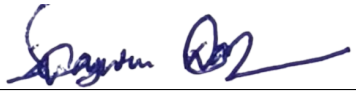
Dr. Swagatam Das
Professor
Electronics and Communication Sciences Unit

Indian Statistical Institute, Kolkata
June, 2024

CERTIFICATE

This is to certify that the dissertation entitled "**Learning with Long-Tailed Noisy Labels**" submitted by **Sarbajit Dey** to the Indian Statistical Institute, Kolkata, in partial fulfillment for the award of the degree of Master of Technology in Computer Science is an authentic and genuine record of the research work conducted by the candidate under my supervision and guidance. I affirm that the dissertation has met all the necessary requirements in accordance with the regulations of this institute.

June, 2024



Dr. Swagatam Das
Professor
Electronics and Communication Sciences Unit,
Indian Statistical Institute,
Kolkata-700108

Acknowledgement

I would like to express my utmost gratitude to my esteemed advisor, Dr. Swagatam Das, from the Electronics and Communication Sciences Unit at the Indian Statistical Institute, Kolkata. His unwavering guidance, continuous support, and encouragement have been invaluable to me throughout this journey. Under his mentorship, I have learned the art of conducting thorough and impactful research, and his insightful ideas have constantly motivated and inspired me.

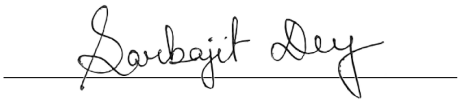
I extend my deepest thanks to Mr. Faizanuddin Ansari, Senior Research Fellow at the Indian Statistical Institute, Kolkata for his invaluable suggestions and engaging discussions, which have significantly helped me and enriched the depth and quality of my work.

I would also like to express my sincere appreciation to all my friends for their assistance and support. I am grateful to everyone who has contributed to my growth and success, even if I have inadvertently missed mentioning them in the above list.

Declaration

I, **Sarbajit Dey**, with Roll No. **CS2226** hereby declare that the material presented in the dissertation titled **Learning with Long tailed Noisy labels** represents original work carried out by me for the degree of **Master of Technology, Computer Science** at **Indian Statistical Institute, Kolkata**.

Furthermore, I affirm that no sections of this report have been sourced or copied from external references without proper attribution. I am aware that any instances of plagiarism or the utilization of unacknowledged materials from third parties will be treated with utmost seriousness and consequences.



Sarbajit Dey
Roll no.- CS2226

Abstract

Deep neural networks (DNNs) have shown exceptionally good performance in a variety of activities by using correctly labelled and 'good' training datasets. These remarkable results, however, are mostly observed with datasets that are carefully controlled and precisely structured. Conversely, data obtained from real-world applications frequently encounter substantial problems that are not commonly found in these 'good' datasets. Two common biases frequently found in real-world data are: (i) long-tailed class distribution, where a small number of classes have a significant number of instances while the rest have only a few, and (ii) label noise, which refers to inaccuracies and errors in the assigned data labels.

When learning models are specifically built to address only one of these biases, either by focusing on the long-tailed nature of the data or on the noise in the labels, their performance declines when they come across data that has both long-tailed distribution and noisy labels, which is a very common occurrence in real-world applications.

This work investigates the complex issue of learning from datasets with long-tailed label noise. In real-world problems such as autonomous driving, medical diagnosis, and large-scale user-generated content platforms, the data obtained frequently shows these properties. Therefore, it is essential to create strong learning algorithms that can successfully address both problems at the same time.

Our objective is to study and make meaningful contributions to the progress of deep learning methods that can effectively handle real-world data difficulties while being robust and dependable. We study the current methods for handling these learning problems, focuse on their shortcomings and try to improve the same.

We propose **Median of Means for centroid estimation** on a clean subset of the dataset. We then use the SFA framework and Semi supervised learning for classification task on imbalanced noisy labels.

Keywords: Classification, class imbalance, Noisy labels, SFA, Median of Means.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Deep Neural Networks (DNN) have demonstrated outstanding performance in a wide range of activities by using large carefully curated and correctly labelled training datasets. These remarkable results, however, are mostly observed with these datasets that are carefully controlled and precisely structured. Conversely, data obtained from real-world applications frequently encounter substantial problems that are not commonly found in these 'good' datasets. Two common biases frequently found in real-world data are: (i) long-tailed datasets, and (ii) noisy labels.

(i) Long-tailed data: Long-tailed data or Imbalanced data refers refers to instances where there is a significant difference in the amount of samples between different classes. The number of samples is very small in most of the classes and significantly higher in some classes.

Within these datasets, the majority class is defined as the class that contains the highest number of samples, while the minority class is characterized by having the lowest number of samples, and he medium classes fall in between these two classes. The imbalance ratio in a particular dataset is defined as the ratio between the cardinality of largest and smallest class. Datasets with high imbalance ratios exhibit poor generalisation on the minority classes. The problems encountered with learning such datasets can be summarised as follows:

Creating precise models becomes challenging when certain classes have very low representation. The low number of samples poses a difficulty in learning and features in the minority classes because there is simply not enough data to learn from.

Another obstacle is to understand the important features for each class. Imbalanced classes make it more challenging to identify the features that are essential for the accurate prediction of one class from another class.

Evaluation metrics might exhibit bias when used to imbalanced data, while being originally created for balanced datasets. This bias results in inaccurate performance evaluations, as the measures are not sufficient to understand the model's capacity to predict the minority class.

Imbalanced datasets occur naturally in the real world like credit card frauds, various disease in the medical field, census data etc.

(ii) Noisy labels: Noisy labels in a dataset refers to data points that are wrongly marked and their labels does not exhibit the ground truth.

There are different types of noise. uniform noise refers to noise distributed evenly over classes. class dependent and feature dependent noise are noise that depends on the class label or feature of the data respectively. Noise ratio is the ratio of noisy samples from all the samples.

In presence of label noise the model exhibits poor performance. The noisy labels have high loss and so direct the gradient abnormally. Whereas over parameterized DNNs tend to learn the noisy labels through memorization and can have perfect training accuracy but this can not be generalised in the validation or test stage.

Noisy labels occur due to problems in data collection or human error or suboptimal methods to collect data. Noisy labels are prevalent in medical data, data collected from internet etc.

Problem Definition:

Consider the classification task with K classes. We denote the training dataset as $D = \{(x_i, y_i)\}_{i=1}^{N}$, where $x_i \in \mathbb{R}^d$ represents the i-th training sample and $y_i \in \{1, 2, \ldots, K\}$ is the corresponding label. For each class $k \in [K]$, we have the training samples as $D_k = \{(x_i, y_i) \mid y_i = k\}$. For this learning task, the training dataset D is long-tailed with the imbalance ratio defined as

$$\rho = \frac{\max_k |D_k|}{\min_k |D_k|},$$

here $\rho \gg 1$, for example $\rho = 100$. So the distribution of the classes is an imbalanced distribution.

Additionally, with the presence of label noise, a fraction of the training samples are mislabeled. That is, there exist some $i \in [N]$ such that $y_i \neq y_i^*$, where y_i^* denotes the ground-truth label for x_i. The noise ratio, that is the fraction of incorrectly labeled training samples is denoted by $\gamma \in (0, 1)$. Here we assume symmetric noise. Given the training dataset D, our objective is to learn a classifier, i.e. the function $f : \mathbb{R}^d \to [K]$ that can generalize well to unseen data for prediction.

Chapter 2

Related Work

As discussed in the previous section, using Deep Neural Networks for learning tasks on imbalanced data with noisy labels result in poor generalisation. Any dataset can be seen as a collection of points in a high dimensional Euclidean space. In classification problems the task for any neural networks is essentially to project the data in an n-dimensional space called the feature space, and learn the decision boundaries in the feature space to separate the classes. The boundaries should be learnt such that the unseen data points can be classified correctly by the network.
various techniques have been proposed over time to deal with imbalanced data and noisy data. Some of the methods for learning with imbalanced data includes . Some of the methods for learning with noisy labels are . However when learning keeping in mind only either label noise or class imbalance the performance deteriorates significantly in real life scenarios where both of the biases are present. Some methods that try to tackle both of these includes HAR[1], PCL[2], RoLT[3].
We study these methods, do a comparison study and try to improve on the results by focusing on the shortcomings of the existing methods. In the following sections we first discuss methods to handle long-tailed distributions, then we discuss methods to tackle label noise and finally discuss some methods that try to tackle both these problems.

2.1 Long Tailed distribution Learning

There are various proposed methods for long tailed learning which can be broadly classified into three categories by the component of the model it modifies- (i) The inputs given to the model, (ii) The outputs generated from the model, The internal structure of the model.

2.1.1 Modifying the Inputs

Popular approaches focus on re-balancing the training data to achieve better representation from minority classes.

Class aware sampling

In order to improve the training process, we employ a sample approach called class-aware sampling. The main objective of this technique is to guarantee that each mini-batch contains a consistent representation of all classes to the greatest extent possible. In addition, this technique ensures that no particular example or class consistently maintains the same order during training.
Practically, we keep two categories of lists: a collective list for classes and separate lists for each class's images. For example in CIFAR-100 dataset we keep a single class list and 100 image lists, each associated with a particular class.
When creating a training mini-batch in an iteration, we begin by randomly choosing a class from the list of classes. Subsequently, we employ a random selection process to choose an image from the list of images associated with the class. In order to preserve diversity and prevent the occurrence of recurrent patterns, we execute a shuffling operation when we reach the end of the list of images belonging to class X. This shuffling rearranges the photos within the designated class. Similarly, as we reach the final class in the list, we rearrange the sequence of the classes.
By utilizing this class-aware sampling technique, we successfully tackle the problem of uneven class distribution. This approach guarantees that the training procedure gains advantages from a more equitable distribution of classes, hence improving the model's capacity to generalize across minority classes.

Dynamic meta-embedding[4]

Our method of dynamic meta-embedding combines a direct image feature with a corresponding memory feature. The feature's norm serves as an indicator of the model's familiarity with known classes. Let's consider a convolutional neural network (CNN) specifically created for classification purposes, which commonly incorporates a softmax output layer. In this type of network, the second-last layer extracts features, while the last layer acts as a linear classifier. These features are trained together utilizing extensive data in a seamless manner. Let v_{direct} represent the direct feature retrieved from an input image. The classification's total accuracy is highly dependent on the quality of this specific attribute.
While feed-forward CNN classifiers show impressive performance with extensive training datasets, they often struggle to get satisfactory results with smaller datasets, due to lack of training data. In order to resolve this problem, we suggest improving

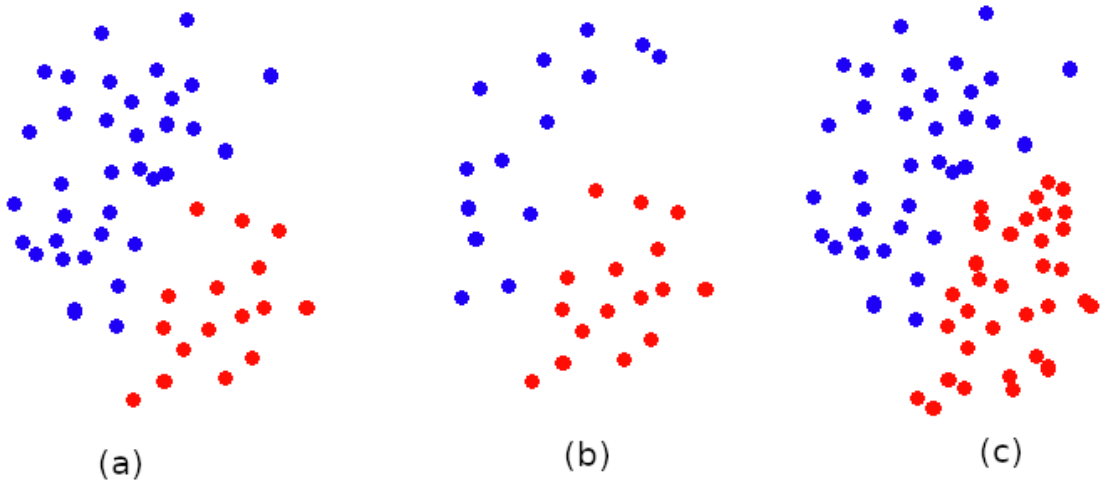


Figure 2.1: (a)Original dataset. Effect of (b) undersampling and (c) oversampling on a dataset.

the direct functionality by including a memory component. This memory function retrieves and stores interconnected visual concepts within a memory module. This concept is similar to the memory techniques used in meta-learning frameworks.
The combined feature is referred to as the meta-embedding v_{meta}, which is then inputted into the final classification layer. The memory feature v_{memory} and the meta-embedding v_{meta} are both derived from the direct feature v_{direct}, enhancing the representation by enhancing the direct feature of the minority classes.

Random oversampling

Random oversampling is a quite simple yet effective technique to address class imbalance. A fixed proportion of samples are randomly selected from the minority classes and added to the training set. Essentially, the minority examples are duplicated and this helps to alleviate the imbalance to some extent. The proportion of data points to be duplicated depends upon the imbalance ratio the user desires to achieve. Random oversampling can, however, lead to overfitting on the minority classes as the same data points make the model memorise the data and unable to generalise well to unseen points.

Random undersampling

Random undersampling is similar to random oversampling, randomly chosen points from the majority class are deleted till the desired imbalance ratio is achieved. Too much undersampling deletes data points possessing valuable discriminatory information, leading to underfitting. The model may struggle to capture the patterns

and characteristics of the majority class. Generally, oversampling performs better than undersampling.

In addition, there are other sampling-based methods like class-balanced sampling where probability of including a point in the training set is proportional to the inverse of the number of samples in the class, square root sampling (sampling probability proportional to inverse square root of class frequency) etc.

2.1.2 Modifying the outputs

All the available methods to address class imbalance we encountered in the literature can be broadly classified into loss-based methods and data-augmentation methods. Loss-based methods focus on modifying the loss function or adjusting the training process to mitigate the impact of class imbalance. They assign different weights or penalties to different classes, thus encouraging the model to pay more attention to underrepresented classes. Whereas, data-augmentation methods increase the variety and quantity of training examples by applying various transformations or generating synthetic examples.

τ-normalised classifier[5]

This method works by readjusting the decision bounds of classifiers. This technique is based on an empirical evidence that when we train using instance-balanced sampling, there is a link between the norms of the classifier weights ($\|w_j\|$) and the size of the classes (n_j). Nevertheless, when the classifiers are adjusted using class-balanced sampling, the magnitudes of the classifier weights tend to become more evenly distributed.

Based on these results, the imbalance in decision boundaries is corrected by directly modifying the norms of the classifier weights using τ-normalization. Let $W = \{w_j\}$ be a matrix in $\mathbb{R}^{d \times C}$, where each w_j is a vector in $\mathbb{R}^d$ representing the classifier weights for class j.

The weights in W are scaled to obtain $W_f = \{w_j^f\}$ through the process of normalization as follows: The formula calculates the value of w_i^f by dividing w_i by the magnitude of w_i raised to the power of τ. Here, τ represents a hyper-parameter and $\|\cdot\|$ represents the L2 norm. When the value of τ is equal to 1, the process of normalizing simplifies to the conventional L2-normalization. On the other hand, when $\tau = 0$, there is no application of scaling. To achieve a smooth change of the weights, the value of τ is empirically selected from interval $(0, 1)$.

After applying the τ-normalization, the classification logits are calculated using the formula:

$$\hat{y} = W_f^\top f(x; \theta).$$

In this context, the symbol $\hat{y}$ represents the predicted logits. The notation $W_f^\top$ refers to the transpose of the adjusted weight matrix, and $f(x;\theta)$ represents the feature representation of the input x that is parameterized by θ.

Post-hoc logit adjustment[6]

Given a long tailed data we learn using a neural network by calculating the logits $f_y(x)$. If we have the logits we generally predict the label as

$$\hat{y} = \arg\max_{y\in[L]} \left(f_y(x)\right).$$

We train the model with softmax and the resultant probabilities can be considered as approximation of underlying probabilities of a data point x belong to a respective class. In this logit adjustment the class is predicted as

$$\hat{y} = \arg\max_{y\in[L]} \left(f_y(x) - \tau \cdot \log \pi_y\right),$$

where π_y are class frequencies of the training sample. τ is a hyper parameter that can be considered as tuning parameter. This method essentially re-weights the output logits to put more focus towards the minority class samples.
There are also momentum based approaches and thresholding approaches applied to the outputs of the model to learn long tailed data.

2.1.3 Other approaches

There are various other methods which are done mostly by modifying the internal design of the model like loss functions of the model or by augmentation methods by modifying the input data presented to the neural network. In loss based approaches the loss function is modified so that the loss from samples in the minority class are heavily weighted to ensure the gradient related to them is updated aggressively. Data augmentation methods are complementary to loss based methods. Unlike loss based methods, data augmentation methods are applied to the input while preparing to feed it into the neural network. Essentially, such methods expand the training set by introducing more novel examples derived from the available data.

Weighted Cross Entropy[7]

The most obvious modification to the simple cross-entropy loss function is to weigh each term in the loss function by the importance of the corresponding class.

$$Loss_{BCE} = -\sum_{i=1}^{N} w_i y_i \log(p_i),$$

where

- w_i represents the weight assigned to class i.
- N is the number of the classes
- y_i is the indicator of the true label of the predicted sample and p_i is the predicted probability for the i^{th} class

This way by introducing class weights, reweighed cross entropy encourages the model to give more emphasis to the minority class during training, helping to improve its performance on underrepresented classes. Generally, the weight w_i is taken $1/N_i$ where N_i is the number of training examples of class i in the training data. A popular approach is also to use the negative log of balanced softmax function with weight w_i as N_i.

Class Balanced Re-Weighing (CBRW)[8]

. In CBRW research paper. It also involves assigning different weights to the samples during training based on their class frequencies.
The reweighing process is based on the inverse of each class' '*effective*' number of samples. The effective number of samples for a class is defined as a weighted average of the actual number of samples and a hyperparameter β. The parameter β controls the degree of reweighing and can be adjusted to control the balance between the minority and majority classes.
The authors assume that each data point occupies a 'volume' in the space where they are represented. When a large number of data points are present, they might overlap and their total occupied volume is not necessarily the sum of the volumes of the individual points. Based on a simple argument, they derive the effective number of samples of a class that occupy as much volume as the existing set of points. The effective number of samples E_n is defined as:

$$E_n = \frac{1 - \beta^{n_i}}{1 - \beta},$$

where n_i is the number of samples from the i^{th} class.
They propose to introduce an extra balancing term in the cross-entropy loss function which weighs the loss by the inverse of the effective number of samples of the class. The modified loss function is known as the CBRW loss.

$$Loss_{CBRW} = -\frac{1-\beta}{1-\beta^{n_i}} * \log(p_i).$$

Focal Loss[9]

The focal loss is defined as

$$Loss_{Focal} = -(1-p_i)^{\gamma}\log(p_i).$$

Focal loss introduces two new concepts: the *modulating factor* and the *focusing parameter*. The modulating factor is defined as $(1-p)^{\gamma}$, where p is the predicted probability of the correct class. This factor reduces the contribution of easy or well-classified examples (where p is close to 1) by increasing the modulating factor value. On the other hand, hard or misclassified examples (where p is close to 0) receive less down-weighting, making them more influential in the loss computation.

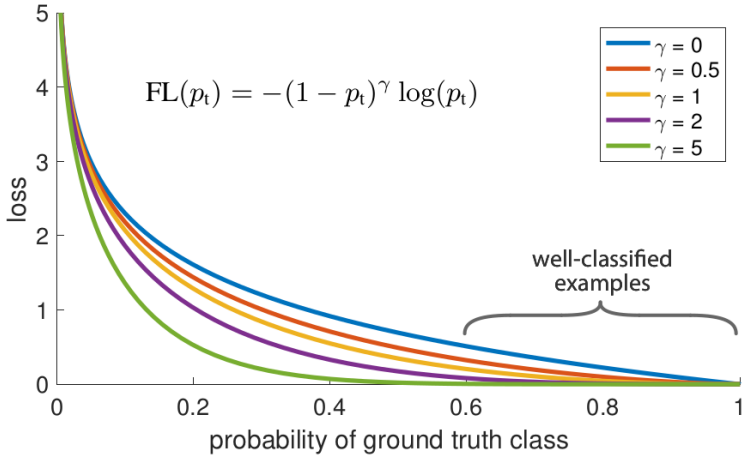


Figure 2.2: As γ increases, Focal Loss concentrates more on hard misclassified examples

The focusing parameter γ is introduced to control the degree of down-weighting. If the data point belongs to the majority class, it is easily classified and p_i is close to 1 for such points. For hard minority class examples, p_i is close to 0. Higher values of γ amplify the effect of the modulating factor, resulting in stronger down-weighting of easy examples. This allows the loss function to focus more on hard examples, which are typically the minority class or objects of interest. The degree of focus for varying values of the focusing parameter γ is shown in figure 2.2.

Through ablation studies, the authors found that $\gamma = 2$ produces the best accuracy results. That's why we also consider the same value of γ in our experiments.

LDAM Loss[10]

The Label Distribution Aware Margin loss, introduced by Zhang et al. is another loss-based method to address class imbalance. To understand LDAM loss, we need to define the class-wise margins of each class. The class-wise margin ξ_i is defined as the minimum distance of a point belonging to the i^{th} class from a decision boundary. Larger margins encourage larger distances between class boundaries, promoting better class discrimination. If we want to classify minority class samples correctly, we would want to encourage the neural network to learn larger margins for the minority classes. But too big margins for the minority classes can make the decision boundary pass through a majority class, and some majority class samples will be incorrectly classified.
There is definitely a trade-off in setting the class margins. Extending a theorem from PAC theory for binary class classification, the authors state that the class margin of a class with n_j samples should be proportional to $1/n_j^4$.
The LDAM loss takes into account both the class-wise margin and the label distribution when computing the loss for each sample. It applies a class-specific margin to the predictions of the model. For a training data point with label j and predicted probability for the i^{th} class is p_i, the LDAM loss is defined as:

$$\text{Loss} = -\log \frac{\exp^{p_j - \Delta_j}}{\exp^{p_j - \Delta_j} + \sum_{i \neq j} \exp^{p_i}}.$$

The loss function is designed to penalise misclassifications of minority samples more heavily, while being more tolerant to misclassifications of majority samples because by definition, it promotes greater class margins for the minority classes.

Mixup[11]

The Mixup technique literally 'mixes' two data points to generate a new data point. Given two data points x_i and x_j with labels y_i and y_j respectively, Mixup generates a new data point $(\bar{x}, \bar{y})$ whose value is $\lambda x_i + (1 - \lambda)x_j$ with label $\lambda y_i + (1 - \lambda)y_j$, where $\lambda \in (0, 1)$. The λ value is drawn from a Beta(α, α) distribution for some value of α.
During training, the mixup procedure is implemented as follows: for each data point in a training batch, another data point is randomly selected from the same batch and 'mixed' with the current point to produce a new data point. In effect,

mixup promotes the idea that points lying on the line joining two data points should also possess a label which is a convex combination of the respective labels. This promotes the neural net to learn smooth decision boundaries between the classes. By creating virtual examples with mixed labels, mixup removes the inherent bias towards majority examples.

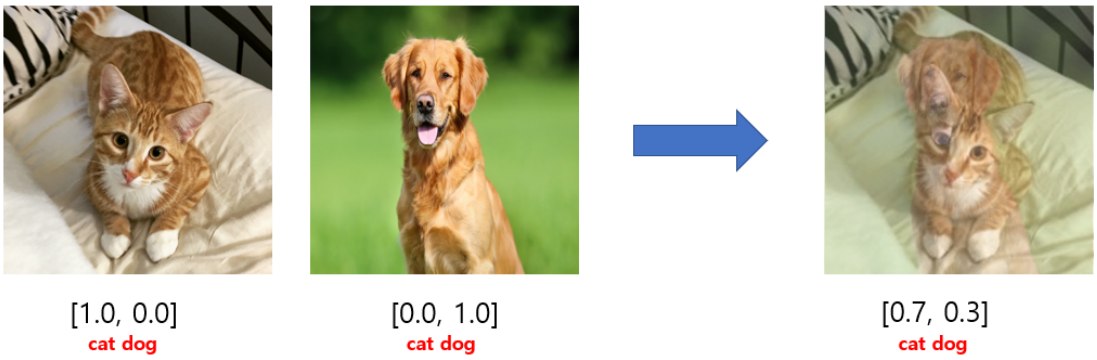


Figure 2.3: Mixup demonstrated between two images of different classes in cat dog dataset

ReMix[12, 13]

ReMix, or Rebalanced Mixup, is similar to mixup but with a slightly different rule. For a pair of data points x_i and x_j, the new data point is given as:

$$\overline{x} = \lambda_x x_i + (1 - \lambda_x) x_j$$
$$\overline{y} = \lambda_y y_i + (1 - \lambda_y) y_j$$

where λ_x is sampled from a Beta distribution as earlier but λ_y is not necessarily equal to λ_x. The value of λ_y is decided by the following rule:

$$\lambda_y = \begin{cases} 0, & n_i/n_j \geq k \text{ and } \lambda < \tau \\ 1, & n_i/n_j \leq 1/k \text{ and } 1 - \lambda < \tau \\ \lambda_x, & \text{otherwise} \end{cases}$$

k and τ are two hyperparameters to be tuned. One way of looking at ReMix is, if the proportion of one class is overwhelmingly more than the other class (the hyperparameter k controls the imbalance ratio between two classes), then assign the new data point a label in favour of the majority class, otherwise a linear combination of the two labels as in Mixup. This prevents the decision boundary from being pushed too far towards the majority class.

CutMix[14]

CutMix performs data augmentation by randomly selecting a portion or patch of an input image and pasting it onto another image. The corresponding labels are also mixed based on the area ratio of the patches. This process encourages the model to learn from both the local and global features present in different images.
It can be seen as a discrete version of mixup where certain portions of the image are completely replaced by another image while other parts remain the same. The area ratio of the patches determines the extent of mixing between the two images. The larger the patch, the stronger the mixing effect. This encourages the model to learn robust features that effectively recognise objects even when the appearance of those objects is altered or occluded. CutMix enables the model to learn locally discriminative features by combining patches from different images. This encourages the model to focus on local patterns and aids in the localisation of objects within an image.

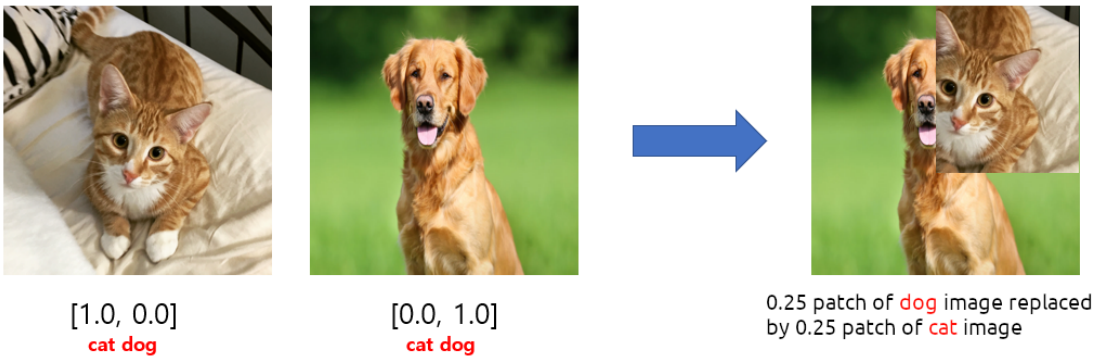


Figure 2.4: Cutmix demonstrated between two images of different classes in cat dog dataset

2.2 Learning with noisy labels

To tackle learning tasks where label noise is present broadly three types of solutions are proposed. (i) Estimating noise transition matrices, (ii) Noise robust loss functions, and (iii) Selecting clean samples from the dataset. We give some overview of each of these methods.

2.2.1 Estimating noise transition matrices

[15][16]The noise transition matrix, is a fundamental concept in the field of label noise learning. The matrix represents the likelihood of accurate labels being transformed into erroneous ones. Formally, the noise transition matrix T is defined as the

relationship between the actual label y and the observed noisy label $\tilde{y}$.The equation is written as: $T_{ij} = P(\tilde{y} = j \mid y = i)$. The conditional probability of $\tilde{y}$ being equal to j given that y is equal to i is denoted by T_ij. The variable T_ij denotes the chance of observing the real label i as label j.
Obtaining an accurate estimation of the noise transition matrix is frequently not feasible because there is no access to the true labels. Instead, it is more practical to approximate this matrix. Here is the method for performing the approximation:
Anchor Points Identification: Identify instances in the dataset that are very probable to have accurate labels. These occurrences, referred to as anchor points, act as dependable benchmarks for assessing the probabilities of noise transfer.
Probabilistic Modeling: Utilize probabilistic models to calculate the probability of witnessing each noisy label in relation to an anchor point. For instance, a classifier that has been trained on a portion of data that is free from errors or a strong model can be used to forecast the likelihoods of labels that are not accurate.
Matrix Construction: Create the noise transition matrix by utilizing the estimated probabilities derived from the probabilistic model. This requires combining the probability estimates from all anchor points.
After approximating the noise transition matrix, it can be included into the learning process to reduce the influence of label noise. There are two prevalent strategies:
Loss Adjustment: Revise the loss function employed during model training to accommodate for the presence of label noise. The revised loss function incorporates the noise transition matrix to adapt the impact of each training instance according to its probability of being accurately classified.
Reallocate the weights of the training samples based on the noise transition matrix. Instances that have a higher probability of being accurately categorized are assigned greater weights, while instances that are more likely to be incorrectly identified are assigned lower weights.

Some papers and methods that applies this method for label noise learning are Gold loss correction, Manifold regularized transition matrix, Forward and backward correction based on estimated stochastic noise matrix.

2.2.2 Noise robust loss functions

The loss based methods focus on employing modified loss functions that have high noise tolerance.

generalised cross Entropy loss[17]

Traditional loss functions, such as the cross-entropy loss, are sensitive to label noise, which can lead to poor generalization and performance. To address this issue, the Generalized Cross Entropy (GCE) loss function has been proposed as a robust alternative.
Cross Entropy Loss- The cross-entropy loss is a widely used loss function in classification tasks, defined as follows for a single training instance:

$$L_{\mathrm{CE}}(p, y) = -\log(p_y),$$

where p_y is the predicted probability for the true class label y. Although effective, the cross-entropy loss can be heavily influenced by noisy labels, leading the model to fit incorrect information.
Generalized Cross Entropy Loss- The Generalized Cross Entropy (GCE) loss generalizes the cross-entropy loss and Mean Absolute Error (MAE), providing a balance between robustness to noise and the ability to learn from clean data. The GCE loss is defined as the negative of the Box Cox transformation:

$$L_{\mathrm{GCE}}(p, y; q) = \frac{1 - p_y^q}{q},$$

where p_y is the predicted probability for the true class label y, and $q \in (0, 1]$ is a parameter that controls the robustness of the loss function. When $q = 1$, the GCE loss reduces to the cross-entropy loss, and as q approaches 0, the GCE loss behaves more like the MAE, which is known to be more robust to noise.
Properties of GCE Loss-

- Noise Robustness: By adjusting the parameter q, the GCE loss can be made more robust to noisy labels. Lower values of q reduce the influence of incorrect labels.
- Flexibility: The GCE loss provides a continuum of loss functions between the cross-entropy loss and the MAE, allowing for flexibility in different noise conditions.
- Smooth Transition: The GCE loss smoothly transitions between different forms, making it adaptable to various levels of noise.

The primary benefits of using GCE loss include improved model robustness to noisy labels, better generalization performance, and flexibility to adapt to different noise levels by tuning the parameter q. By introducing the parameter q, the GCE loss can be adjusted to balance between noise robustness and learning efficacy.

ELR loss[18]

The ELR loss function is designed to prevent the model from overfitting to noisy labels by regularizing the learning process during the early stages of training. This regularization helps the model focus on the correctly labeled data before it starts memorizing the noisy labels.
The ELR loss combines the standard cross-entropy loss with a regularization term that discourages the model from fitting noisy labels. The loss function can be defined as follows:

$$L_{\mathrm{ELR}}(\theta) = -\frac{1}{N}\sum_{i=1}^{N}\left[\log p_\theta(y_i \mid x_i) + \lambda \cdot (1 - \hat{p}_\theta(y_i \mid x_i))\right],$$

where:

- θ represents the model parameters.
- N is the number of training samples.
- (x_i, y_i) are the input and true label for the i-th training sample.
- $p_\theta(y_i \mid x_i)$ is the predicted probability of the true label y_i given the input x_i.
- $\hat{p}_\theta(y_i \mid x_i)$ is the historical probability of the true label y_i given the input x_i, tracked over earlier epochs.
- λ is a regularization hyperparameter that controls the weight of the regularization term.

The regularization term $(1 - \hat{p}_\theta(y_i \mid x_i))$ penalizes the model for assigning high probabilities to noisy labels, thus helping the model to focus on clean labels.
Properties of ELR Loss:

- Noise Resistance: ELR loss is designed to be robust against noisy labels by regularizing the learning process and focusing on early-learned patterns.
- Dynamic Adaptation: By using historical probabilities, ELR dynamically adapts to the model's learning progress, emphasizing correct labels over time.
- Simple Integration: ELR can be easily integrated into existing training frameworks with minimal modifications.

The Early-Learning Regularization (ELR) loss function provides a robust alternative to traditional loss functions for training deep neural networks with noisy labels. By incorporating a regularization term that leverages early-learning patterns, ELR helps prevent overfitting to noisy labels.

2.2.3 Clean sample selection

These methods are based on the fact that not all the samples in the dataset are corrupted with noise and there is a good proportion of clean data available. To leverage the clean samples in the learning we try to separate the clean samples from noisy samples. Two problem arises here (i) How to correctly predict the noisy samples and (ii) After extracting the clean samples what to do with the noisy samples, since if we simply discard them we are essentially losing data. various methods are proposed for label noise learning using this approach.

Co teaching[19]

Co-teaching is a powerful approach specifically developed to reduce the negative impact of inaccurate labels. The main concept is to concurrently train two neural networks. Instead of individual networks relying solely on their own projected labels, they share information and learn from each other's more accurate instances. This collaborative learning method assists in recognizing and reducing the consequences of inaccurate labels.
(1) Initial Training and Divergence: Commence by initializing two neural networks, f and g, with distinct random weights. Both networks are trained using the identical noisy dataset for a limited number of epochs. This preliminary stage helps in understanding the distribution of the data, while simultaneously ensuring that the networks begin to diverge in their predictions.
(2) During the training process, each network chooses a selection of examples that have the lowest losses based on its own forecasts. The underlying assumption is that instances with lesser losses are more prone to being accurately classified. The network f chooses a subset D_f of examples with the lowest losses from its training set. Similarly, the network g chooses a subset D_g of examples with the lowest losses from its training set.
(3) Each network updates its parameters by including the examples chosen by the other network. More precisely, the function f is updated using the information from D_g, whereas the function g is updated using the information from D_f. This interaction helps in mitigating the bias caused by inaccurate labels as one network depends on the more dependable instances indicated by the other.
(4) This process of repeatedly choosing samples with minimal loss and adjusting parameters is continued iteratively. Over time, both networks develop more resilience to noisy labels, as they consistently learn from cleaner subsets of the data.
Benefits of Co-Teaching includes-
Noise Robustness: The networks enhance their ability to withstand noise by utilizing the clean subsets from each other.

Variety of Learning Trajectories: The networks, initialized with varying parameters, explore a variety of learning trajectories, which aids in encompassing a wider range of the data distribution.
Mitigated Overfitting: The utilization of the small-loss method guarantees that the neural networks prioritize more dependable instances, hence diminishing the likelihood of overfitting to erroneous labels.
Typical cross-entropy loss function is generally used. Though, adjustments can be done to more effectively learn label noise, such as integrating a loss function that is robust to noise.
The choice of the proportion of small-loss samples, represented by a hyperparameter τ, is of utmost importance. The starting level is usually high and subsequently falls, enabling the networks to first process a larger number of samples and gradually prioritize the most dependable ones.

Various modifications and alternatives to co-teaching have been suggested in order to enhance its effectiveness:
Co-Teaching plus: A more advanced iteration in which each network improves the chosen examples by comparing them with the predictions of the other network.
Decoupling: It is a method similar to co-teaching, but it uses distinct processes to divide and utilize the data for training.
Co-Regularization: Implements regularization methods to prevent excessive divergence in the learning paths of both networks.
In conclusion Co-teaching provides a novel approach for learning with noisy labels by utilizing the cooperative training of two neural networks. The iterative and mutual learning process greatly improves the model's capacity to generalize even in the presence of noisy data. As research advances in this field, co-teaching is evolving, including tactics and variations to further increase its applicability.

DivideMix[20]

DivideMix addresses the issue of label noise learning through a combination of semi-supervised learning and Gaussian Mixture Models[21] (GMMs).
The dataset is first divided two groups: one with clean labels and one with noisy labels. This is done using a GMM, which models the distribution of the loss values (the differences between the predicted and actual labels) for each sample. Samples with lower loss values are more likely to be clean, while those with higher loss values are more likely to be noisy. This is also called the small loss criterion.
After splitting the data, DivideMix uses semi-supervised learning to make use of all available data. Clean samples are treated as labeled data, and noisy samples are treated as unlabeled data. This approach helps in utilizing the information

present in noisy samples without being directly affected by their incorrect labels. For the noisy samples, DivideMix generates refined labels using MixMatch, which combines weakly augmented versions of the data and computes a consensus label. This guessed label is then used to help guide the learning process.
further enhance robustness, DivideMix employs a co-training strategy. It trains two separate neural networks simultaneously. Each network's predictions are used to help refine the labels for the other network, which helps in reducing the bias that might arise from using a single network. It also incorporates consistency regularization and strong data augmentation techniques. Consistency regularization ensures that the model's predictions are stable under small input perturbations, while data augmentation helps in making the model robust to variations in the input data.

2.3 Imbalanced noisy data

There has been some prior works on this field. We look at some of the popular methods.

2.3.1 RoLT

Robust long-tailed learning (RoLT)[3] is designed to handle the dual challenges of long-tailed class distributions and noisy labels.
RoLT incorporates a sample selection mechanism that dynamically identifies and separates clean samples from noisy ones. This mechanism is particularly tuned to handle imbalances, ensuring that minority classes are adequately represented during training. By identifying and isolating noisy samples, RoLT can focus the learning process on cleaner data, which helps in maintaining the robustness of the model. This often involves the use of loss metrics and confidence scores to determine the likelihood of a sample being clean or noisy. It uses a class-balanced loss function that compensates for the long-tailed distribution of classes. This ensures that the learning process does not overly favor the majority classes, which can happen with standard loss functions such as cross entropy loss. Consistency Regularization techniques are used to ensure that the model's predictions remain stable under different augmentations of the input data.
A Gaussian Mixture Model (GMM) that models the distribution of distance of features from the class prototype values is used to separate clean samples from noisy samples. As discussed above clean samples generally have lower loss values, while noisy samples exhibit higher losses. After the initial separation, RoLT treats clean samples as labeled data and noisy samples as unlabeled data. Using semi-supervised learning techniques, RoLT can effectively utilize the information present in noisy

samples. A class balanced loss function is used and consistency regularization is applied.
By effectively combining several advanced techniques to handle long-tailed distributions and noisy labels and leveraging semi-supervised learning, class-balanced loss functions, and consistency regularization, RoLT can train robust models that generalize well even in long-tailed noisy datasets.

2.3.2 HAR

Heteroskedastic adaptive regularization (HAR) [1] deals with heteroskedastic noisy in imbalanced datasets. In the imbalanced setting while trying to detect noise based on small loss criterion the rarely occurring samples might get detected as noise because of having large error. HAR employs a regularization technique based on the Lipschitz regularizer and finds the optimal level for the regularization. The higher the uncertainty of a sample point the stronger the regularization applied. This method helps in learning with imbalanced and noisy data combined.

2.3.3 RCAL

Representation calibration method or RCAl[22] uses the features or deep representation of the model and tries to recover the original representation before the introduction of the noise. The model first employs contrastive learning techniques to get unsupervised representation of the samples. It then tries to recover the original Gaussian distribution of the classes. Also information on the distribution of the majority classes are used to calibrate the minority class estimations. This calibration is done since there are not significant data points in the rare classes and that can result in the estimated distribution being biased.

2.3.4 SFA

Stochastic feature Averaging or SFA[23] employs Bayesian inference with a Gaussian approximation to the posterior distribution over class centroids. First at each epoch the immediate centroids for the feature vectors of each class is calculated. To ensure that over time this estimation becomes more robust and unbiased a moving average for the class centroids and their second order mean is maintained. So the posterior distributions for the class centroids will be normally distributed along with the means and standard deviation calculated from the above calculated first and second order moments.
Let, $C_{SFA} = [c_1, ..., c_K]$ represents the running estimate of the class centroids, C_t denotes the instantaneous centroid estimates (calculated in the previous subsection)

after the t-th training epoch, and α is the smoothing factor. Then,

$$C_{SFA} = \alpha \cdot C_{SFA} + (1 - \alpha) \cdot C_t.$$

$$C^{(2)}_{SFA} = \alpha \cdot C^{(2)}_{SFA} + (1 - \alpha) \cdot C^2_t,$$

Here, C^2 refers to the element-wise square.

Chapter 3

Our contribution

When learning with long-tailed noisy labels, the many methods specifically developed for imbalanced or noisy data often exhibit subpar performance. Therefore, it is crucial to develop algorithms that are capable of handling classification jobs that involve both noisy labels and long-tailed class distribution. We analyze these methods and do a comparative analysis on them. Subsequently, our attention is directed towards the constraints of these methodologies, with the aim of enhancing the current outcomes.

3.1 A few preliminaries

3.1.1 Gaussian Mixture Model (GMM)

A Gaussian Mixture Model (GMM)[21] is a statistical model that assumes that all the data points are derived from a combination of a limited number of Gaussian distributions with parameters that are not known. Its versatility in modeling complex data distributions makes it highly useful in clustering, density estimation, and pattern recognition problems.

Gaussian Distribution

A Gaussian distribution in d-dimensions is defined by the equation:

$$\mathcal{N}(x|\mu,\Sigma) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right),$$

where:

- x represents a data point with d dimensions.
- μ represents the vector of means.

- Σ represents the covariance matrix.
- $|\Sigma|$ represents the determinant of Σ.

Mathematical Formulation

A Gaussian Mixture Model (GMM) is a mathematical model that may be expressed as the sum of K Gaussian component densities, each weighted by a certain factor. The GMM's probability density function is defined as the sum of the product of the weight coefficients and the Gaussian distribution for each component. It may be expressed as:

$$p(x) = \sum_{k=1}^{K} \phi_k \mathcal{N}(x|\mu_k, \Sigma_k),$$

where:

- ϕ_k represent the mixture weights,
- $\sum_{k=1}^{K} \phi_k = 1$ and $\phi_k \geq 0$.
- $\mathcal{N}(x|\mu_k, \Sigma_k)$ represents the Gaussian (normal) distribution with a mean of μ_k and a covariance matrix of Σ_k.

The Expectation-Maximization (EM) algorithm is used to estimate the parameters of a Gaussian Mixture Model (GMM). The process iteratively adjusts the parameters in order to optimize the likelihood of the observed data. Deciding on the number of components K and initializing the parameters are vital stages in the process of fitting a Gaussian Mixture Model (GMM). Several methods for initializing the model are provided: (i) Initialize the means μ_k using the K-means algorithm. (ii) Select K data points as beginning means in a random manner. (iii) Employ hierarchical clustering to establish initial cluster assignments.
Gaussian Mixture Model (GMM) can be used for clustering technique, where each Gaussian component represent a cluster. Gaussian Mixture Models (GMM) have the advantage over K-means in that they can accurately represent elliptical clusters and provide soft cluster assignments (probabilistic memberships).
GMM is also employed for estimating the probability density function of a dataset, as well as for tasks involving pattern recognition, such as speech recognition and image segmentation.
Gaussian Mixture Models (GMM) have the ability to represent complex probability distributions and are not constrained to clusters with spherical shapes.Gaussian Mixture Model (GMM) enables the application of soft clustering, which provides further insights into the cluster memberships of data points. Although the process of fitting a Gaussian Mixture Model (GMM) can be computationally demanding,

the results can be influenced by the initialization method used, and determining the appropriate number of clusters may not be a straightforward task. Here we use a GMM with two components.

3.1.2 Balanced Classifier

[7]As discussed in the previous chapter in the field of long-tailed learning, class rebalancing strategies are essential for addressing the imbalance data that arise when dealing with datasets where some classes are significantly underrepresented compared to others. These methods play a crucial role in improving overall classifier performance and representation learning with respect to the original data distribution. Previous research has highlighted the benefits of decoupling the processes of representation learning and classifier learning. This decoupling approach has demonstrated superior results compared to conventional learning methods. To enhance performance in the presence of class imbalance, we implement a two-branch network during model training. This network introduces an auxiliary classifier to the neural network's backbone, which is designed for balanced classifier learning. At the same time, the original classifier is retained for effective representation learning. The two classifiers are denoted as f_{abc} and f respectively.
The classifier f is trained using the standard cross-entropy loss function defined as:

$$\ell_{\text{CE}}(x, y) = -\log \frac{e^{z_y}}{\sum_{k=1}^{K} e^{z_k}},$$

where $z = f(x)$, representing the logits produced by the classifier f.
Balanced Softmax Loss for Auxiliary Classifier: To produce unbiased predictions, the auxiliary classifier f_{abc} is trained alongside the standard classifier f by minimizing the Balanced Softmax (BS) loss function. This classifier is integrated into the network to share the same feature extractor as f. The Balanced Softmax loss is defined as:

$$\ell_{\text{BS}}(x, y) = -\log \frac{n_y e^{z_y}}{\sum_{k=1}^{K} n_k e^{z_k}},$$

Here, n_k denotes the number of samples in class k counted from the dataset D, and z_k represents the k-th logit produced by $f_{\text{abc}}(x)$.
The key idea behind the Balanced Softmax loss is to adjust the loss contributions based on the class frequencies. Specifically, it penalizes the majority class samples more heavily while reducing the penalty on minority class samples. This approach helps to create a more balanced classifier, mitigating the bias towards the majority class that typically arises in imbalanced datasets.
The inclusion of the auxiliary balanced classifier f_{abc} addresses the class imbalance by modifying the training dynamics. The Balanced Softmax loss ensures that the

classifier pays more attention to underrepresented classes by scaling the loss inversely with the class frequencies. This mechanism allows the model to learn a more balanced decision boundary, improving its generalization across all classes.
The main classifier f continues to be trained with the standard cross-entropy loss, ensuring that the feature representations extracted by the neural network are not biased by the re-balancing adjustments. By decoupling the tasks of representation learning and classifier adjustment, this two-branch network architecture leverages the strengths of both approaches, enhancing the overall performance on imbalanced datasets.

3.1.3 MixMatch

MixMatch[24] is a semi-supervised learning algorithm designed to efficiently utilize both labeled and unlabeled data to improve the performance of machine learning models. It achieves this by combining several key techniques: data augmentation, label guessing, and mixup. We study the MixMatch algorithm and its components in details.

Key Concepts and Techniques

Data Augmentation: Data augmentation involves generating new training examples by applying random transformations to existing data. This helps to increase the diversity of the training set and improve the generalization of the model.
Label Guessing: For unlabeled data, MixMatch generates "guessed" labels by averaging predictions made by the model on different augmentations of the same unlabeled example. This process provides a proxy for the true labels of the unlabeled data.
Mixup: is a technique that creates new training examples by interpolating pairs of examples and their labels. This encourages the model to behave linearly between training examples, improving its robustness.

Algorithm

The MixMatch algorithm combines these techniques into a coherent training procedure:
Input:

- X_l: Labeled data with labels.
- X_u: Unlabeled data without labels.
- α: Mixup parameter controlling the interpolation strength.

- T: Temperature parameter for sharpening the guessed labels.

Augmentation and Label Guessing:

- For each mini-batch, perform data augmentation on both labeled and unlabeled data.
- Apply the model to multiple augmentations of each unlabeled example to generate multiple predictions.
- Average these predictions to create soft pseudo-labels for the unlabeled data.
- Apply a sharpening function to these soft labels to produce more confident pseudo-labels:

$$\text{sharpen}(q_i, T) = \frac{q_i^{1/T}}{\sum_j q_j^{1/T}},$$

  where q_i are the soft labels and T is the temperature.

Mixup:

- Combine the labeled and pseudo-labeled data.
- Apply the mixup operation to generate new training examples:

$$\tilde{x} = \lambda x_i + (1 - \lambda) x_j,$$

$$\tilde{y} = \lambda y_i + (1 - \lambda) y_j,$$

  where λ is sampled from a Beta distribution: $\lambda \sim \text{Beta}(\alpha, \alpha)$.

Loss functions:

- Compute the cross-entropy loss on the labeled data.
- Compute the mean squared error loss on the pseudo-labeled data.
- Combine these losses into a single loss function to optimize the model:

$$\mathcal{L} = \mathcal{L}_{\text{supervised}} + \mathcal{L}_{\text{unsupervised}},$$

$$\mathcal{L}_{\text{supervised}} = \frac{1}{|X_l|} \sum_{(x,y) \in X_l} H(\tilde{y}, f(\tilde{x})),$$

$$\mathcal{L}_{\text{unsupervised}} = \frac{1}{|X_u|} \sum_{(x,\tilde{y}) \in X_u} \|\tilde{y} - f(\tilde{x})\|_2^2.$$

Training: Stochastic gradient descent (SGD) or any other optimization method is used to minimize the combined loss function.

Advantages

MixMatch leverages unlabeled data effectively through label guessing and consistency regularization, improving model performance even with limited labeled data. By using mixup, MixMatch encourages the model to generalize better and be more robust to input variations and noise.
The mixup algorithm can be modified and applied to different learning tasks easily. The combination of augmentation, mixup, and label guessing helps the model to generalize well to unseen data.

Conclusion

MixMatch is a powerful semi-supervised learning algorithm that combines data augmentation, label guessing, and mixup to make efficient use of both labeled and unlabeled data. Its ability to improve model performance with minimal labeled data makes it particularly useful in scenarios where labeled data is scarce but unlabeled data is abundant. The algorithm's robustness and generalization capabilities have made it a popular choice for semi-supervised learning tasks.

3.1.4 Median of Means

The Median of Means (MoM)[25] is a robust method for estimating the mean of a distribution, particularly useful when dealing with heavy-tailed data or outliers. In most of the machine learning algorithms to tackle noisy data or imbalanced data an approximation is made that **The feature vectors are normally distributed in the feature space**. However in presence of noisy and imbalanced data this may not hold true. In that case the sample mean won't be an accurate guess for the population mean. So we study other mean estimating methods like MoM.
In MoM we divide the data into several groups, computes the mean of each group, and then takes the median of these means, thus reducing the influence of outliers and providing a more reliable estimate of the mean.

Median of Means Algorithm

Given a sample $X = \{x_1, x_2, \ldots, x_n\}$ of size n, divide the data into k disjoint groups (or batches) of equal size $b = n/k$.
Calculate the mean of each group. Let $\bar{X}_i$ denote the mean of the i-th group. Thus, you obtain k means: $\bar{X}_1, \bar{X}_2, \ldots, \bar{X}_k$.
Compute the median of the k group means to obtain the final estimate of the population mean.

Mathematical Formulation

Let X be a sample from a distribution with unknown mean μ. The sample X is divided into k groups, each containing b observations. Denote the observations in the i-th group as $X_{(i)} = \{x_{(i-1)b+1}, x_{(i-1)b+2}, \ldots, x_{ib}\}$.
The mean of the i-th group is:

$$\bar{X}_i = \frac{1}{b} \sum_{j=(i-1)b+1}^{ib} x_j.$$

The Median of Means estimator $\hat{\mu}_{\text{MoM}}$ is then defined as the median of the $\bar{X}_i$'s:

$$\hat{\mu}_{\text{MoM}} = \text{median}(\bar{X}_1, \bar{X}_2, \ldots, \bar{X}_k).$$

Properties of Median of Means Estimator

Robustness: The MoM estimator is robust to outliers. Even if some groups contain outliers, the median operation ensures that their influence is limited.
Concentration: The MoM estimator has good concentration properties. With high probability, it is close to the true mean μ of the distribution.
Variance: The variance of the MoM estimator is controlled, making it a reliable choice for estimating the mean, especially in the presence of heavy-tailed distributions.
Bias: The MoM estimator can be slightly biased, especially for small sample sizes. However, the bias decreases as the sample size increases.

Conclusion

The Median of Means method provides a robust and reliable way to estimate the mean of a distribution, particularly in the presence of outliers or heavy-tailed data. By dividing the data, computing group means, and taking the median, this method effectively reduces the influence of outliers and improves the accuracy of the mean estimation.

3.2 Proposed Method

We detailed the algorithm used in the following sections. Starting with a warmup stage to get initial representation of the features and get the initial class estimations of the data points. We then use the co-teaching framework to use features from one model to train the other model. We first compute the instant class centroids using our proposed method. To get a better representation of the class centroids over

training epochs we keep a moving average following Stochastic feature averaging. We now use the small distance criteria to separate the clean samples from noisy samples by fitting a Gaussian Mixture model. Once we have the labelled and unlabelled data we can train the model using the MixMatch framework for semi-supervised learning. Finally we test or model on the test dataset to understand the generalization of the model.

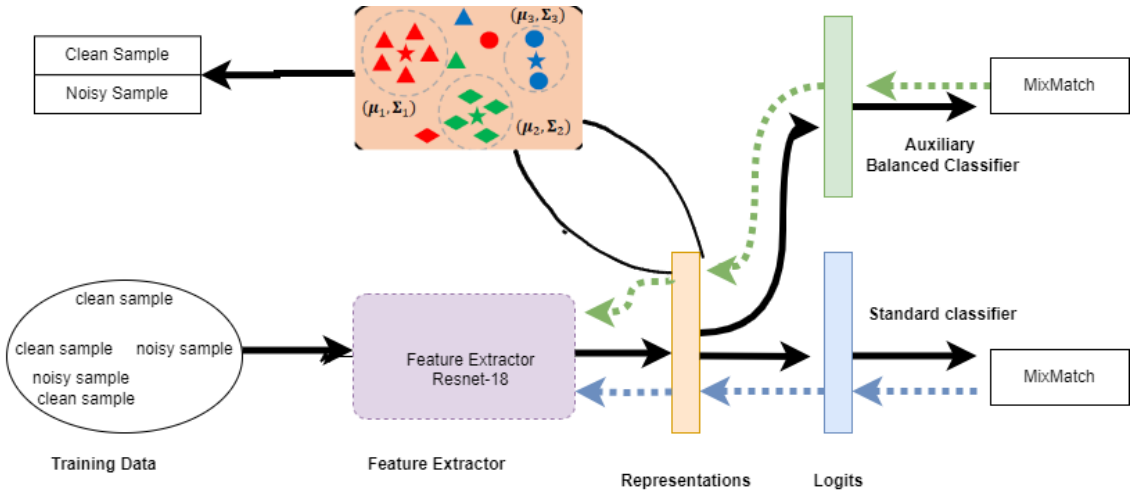


Figure 3.1: Architecture for our model

3.2.1 Instant centroid estimation

A straightforward method for estimating class centroids involves calculating the mean of all feature vectors for each class after every training epoch. This approach, known as instant centroid estimation, can be problematic in the presence of label noise, as it includes noisy samples in the calculation, leading to inaccurate estimates. To address this challenge, it is essential to utilize as many clean samples as possible when computing class centroids. Previous work suggests that both majority and minority classes typically exhibit similar ranges of predicted confidence from the auxiliary balanced classifier. Additionally, clean samples generally show higher confidence levels compared to noisy ones. This insight suggests selecting a subset of clean samples for each class based on a confidence threshold θ_n, to achieve a more accurate centroid estimate.

Specifically, the centroid for class k can be computed as follows:

$$c_k = \text{Normalize}\left(\frac{1}{N_k}\sum_{x \in D_k} \mathbb{I}(f_{abc}(x) > \theta_n) g(x)\right),$$

Here, $\mathbb{I}(\cdot)$ is an indicator function, N_k is the number of selected samples, $g(x)$ returns the latent representation of x, and $f_{abc}(x)$ gives the predicted confidence from the

auxiliary balanced classifier as discussed above.
In practice, we expect that as training progresses, more samples will achieve high predicted confidence. Therefore, we set a dynamic threshold θ_{nt} that increases with each iteration t.
But notice that if sample mean is used to estimate the population mean we use an underlying assumption that the feature vectors are normally distributed which might not be the case. So instead we use robust MoM estimator for computing the mean. So, the centroid of class k is calculated as:

$$c_k = \text{Normalize}\left(\text{MoM}\left(\mathbb{I}(f_{abc}(x) > \theta_n)g(x)\right)\right).$$

By using this method of instant centroid estimation, we can derive relatively reliable centroids for selecting clean samples. However, the precision of the estimate may still be influenced by the limited number of samples in minority classes.
The robust class centroid estimation method mitigates the effect of label noise by focusing on high-confidence clean samples, dynamically adjusting thresholds to improve accuracy throughout the training process. This approach ensures more reliable centroid calculations, crucial for effective learning, especially in imbalanced datasets with noisy labels.

3.2.2 Final centroid estimation

$$C_{MoM} = \alpha \cdot C_{MoM} + (1 - \alpha) \cdot C_t,$$

Here, $C_{SMoM} = [c_1, ..., c_K]$ represents the running estimate of the class centroids, C_t denotes the instantaneous centroid estimates (calculated in the previous subsection) after the t-th training epoch, and α is the smoothing factor. This averaging process in the latent representation space captures the dynamic nature of training in deep neural networks (DNNs) and provides a robust estimate of the centroid.
Running average of the second moment for class centroids are:

$$C^{(2)}_{MoM} = \alpha \cdot C^{(2)}_{MoM} + (1 - \alpha) \cdot C^2_t,$$

Here, C^2 refers to the element-wise square. Thus, a diagonal covariance matrix can be approximated by $\Sigma_{MoM} = \text{diag}(C^{(2)}_{MoM} - (C_{MoM})^2)$. Consequently, the posterior distribution over the class centroid can be represented as a Gaussian distribution $N(C_{MoM}, \Sigma_{MoM})$.

3.2.3 Choosing clean samples

Since we have the posterior distributions we now sample $\tilde{C}$ from $N(C_{MoM}, \Sigma_{MoM})$. Then we calculate the Euclidean distances between the estimated class centroid $\hat{c}_k$ and samples of class k, we use the following formula:

$$\text{dist}(\tilde{c}_k, x_i) = ||\tilde{c}_k - g(x_i)||_2^2.$$

This process is repeated S times, and the average distance is computed. As a result, the training samples are well-clustered based on their distances to the class centroid. These distances are then modeled using a two-component Gaussian mixture model (GMM) as discussed above:

$$\text{dist} \sim \sum_{j=1}^{2} \phi_j N(\mu_j, \sigma_j^2),$$

where μ_j and σ_j represent the mean and variance of the j-th Gaussian component. Assuming $\mu_1 < \mu_2$, clean samples are expected to cluster around the class centroids, while noisy samples are more dispersed. The probability that a sample x_i is clean is defined as:

$$P(\text{clean} \mid x_i) = \frac{\phi_1 N(\mu_1, \sigma_1^2)}{\sum_{j=1}^{2} \phi_j N(\mu_j, \sigma_j^2)}.$$

Samples with $P(\text{clean} \mid x_i) > 0.5$ are considered clean, while the rest are categorized as noisy. The training dataset D is thus divided into a clean sample set D_{clean} and a noisy sample set D_{noisy}.
So we get two subsets of the training data as labelled data and unlabelled data.

3.2.4 Overall Algorithm

The overall algorithm combined with all the different components discussed in this section is given in **Algorithm 1**.

Algorithm 1 Training Algorithm

Require: Training dataset $D = \{(x_i, y_i)\}_{i=1}^{N}$, model parameters θ, sampling rate S, warm-up epochs T_0, and total training epochs T.

// Warm-up
for $t = 1, \ldots, T_0$ **do**
  $L = \ell_{CE}(D, f) + \ell_{BS}(D, f_{abc})$
  $\theta_t = \text{SGD}(L, \theta_{t-1})$
end for

for $t = T_0 + 1, \ldots, T$ **do**
  // Sample selection
  $D_{\text{clean}} = \emptyset$, $D_{\text{noisy}} = \emptyset$
  Compute threshold $\tau_t = \gamma_t \hat{\tau}$
  for $k = 1, \ldots, K$ **do**
    Compute class centroid as in section 3.2.1
    Update C_{MoM} and C'_{MoM} as in section 3.2.2
    Sample $\tilde{c}_k \sim N(C_{\text{MoM}}, \Sigma_{\text{MoM}})$
    $\text{dist}_{i,s} = ||\tilde{c}_k - g(x_i)||^2$, where $x_i \in D_k$
    $D_{\text{clean}_k}, D_{\text{noisy}_k} = \text{GMM}(\text{dist})$
    $D_{\text{clean}} = D_{\text{clean}} \cup D_{\text{clean}_k}$, $D_{\text{noisy}} = D_{\text{noisy}} \cup D_{\text{noisy}_k}$
  end for
  // Semi supervised learning
  $L_{SSL} = \text{MixMatch}(D_{\text{clean}}, D_{\text{noisy}}, f)$
  $L_{ABC} = \text{MixMatch}(D_{\text{clean}}, D_{\text{noisy}}, f_{abc})$
  $L = L_{SSL} + L_{ABC}$
  $\theta_t = \text{SGD}(L, \theta_{t-1})$
end for

Chapter 4

Experimental Results and Observations

This chapter presents the results of our extensive experiments and our observations and interpretation of those results. We initially did a comparative study for the existing state of the art methods for learning with noisy and imbalanced datasets. We have based our model on stochastic feature averaging paper and modified it. We experimented with different values for the hyperparameter k for the number of blocks in the MoM estimation. We have mainly worked with two datasets CIFAR-10 and CIFAR-100 and compared the overall accuracy performance and accuracy performance for majority, and minority classes.

4.1 Imbalanced noisy CIFAR dataset

The CIFAR-10 and CIFAR-100 datasets are widely used benchmark datasets in machine learning and are designed for fine-grained object recognition tasks. The CIFAR-10 dataset has 50000 training data and 10000 test data. There are 5 equal training batches and 1 test batch. There are 10 classes. CIFAR-100 consists of 100 classes, each containing 600 images, resulting in a total of 60,000 images. The dataset is divided into a training set of 50,000 images and a test set of 10,000 images. Each image in CIFAR-10 and CIFAR-100 have a fixed size of 32x32 pixels. The images are RGB (color) images, i.e., they have three color channels (red, green, and blue) for each pixel.
Since CIFAR datasets are commonly used as a benchmark dataset for evaluating the performance of various machine learning , we test our method on this datasets. But our method is designed to help in better learning in a class imbalanced scenario, whereas the CIFAR datasets are inherently balanced in nature. To simulate training data with long-tailed noisy labels, we induce artificial class imbalance and artificial

noise. We have defined the imbalance ratio as ρ and the noise ratio as γ. The number of samples for the k-th class is set to $N_k = \frac{N}{\rho^{\frac{k-1}{K-1}}}$, creating a long-tailed dataset. Label noise is then introduced using a noise transition matrix T, where:

$$T_{ij} = P(Y = j \mid Y^* = i) = \begin{cases} 1 - \gamma & \text{if } i = j \\ \frac{N_j}{N - N_i}\gamma & \text{otherwise} \end{cases}$$

For our experiments, we utilize an 18-layer PreAct ResNet model, For CIFAR-10 dataset we classify the classes with class labe 0, 1 as majority class, the classes with class label 8, 9, 10 as minority class and the rest of the classes as medium class. For CIFAR-100 dataset we classify the classes with at-least 100 data-points as majority class, classes with less than 20 data-points as minority class and the rest of the classes as medium class.

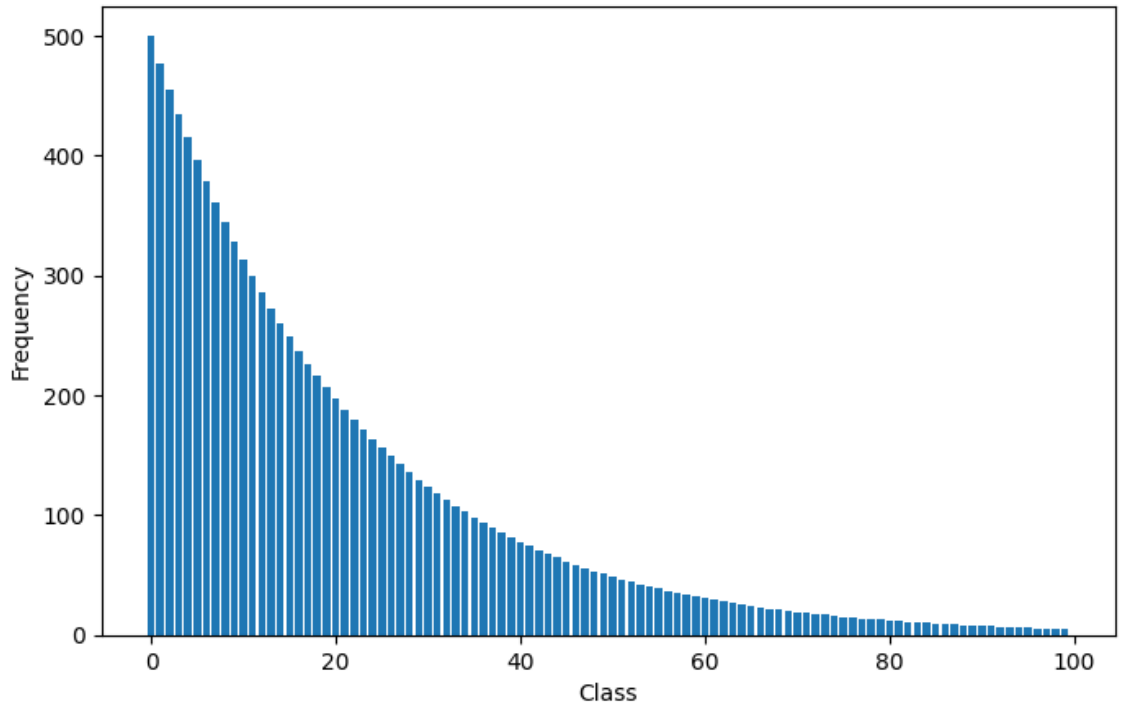


Figure 4.1: Class distribution of data points in the Imbalanced CIFAR100 dataset

4.2 Implementation details

For training purpose, we take a 18-layer PreAct ResNet model. ResNet-18 is a part of the ResNet (Residual Network) family of Convolutional Neural Networks. The basic building block of this family of neural networks is the 'residual block' which has skip connections across layers to promote better flow of information between layers. Some modifications are made to adapt the network to the CIFAR data image size of 32x32 pixels, like the initial 7x7 convolutional layer with stride 2 and

max pooling is replaced with a 3x3 convolutional layer with stride 1 and no max pooling. ResNet-18 has been shown to achieve high accuracy on CIFAR-10 and CIFAR-100 datasets and is widely used as a baseline model for benchmarking image classification performance on these datasets.
Our model is trained using stochastic gradient descent (SGD) with a α of 0.9, a weight-decay of 5×10^{-4}. Our batch size is 128, and an initial learning rate of 0.02. The training process runs for 200 epochs. The model is warmed up for 30 epochs. Learning rate is reduced to 1/10-th of initial learning rate after 150 epochs. In our experiments, we vary the imbalance ratio ρ within {10, 50, 100} and vary the noise level γ within {0.2, 0.3, 0.4, 0.5}.

4.3 Comparison results

The baseline methods for us correspond to the simple cross-entropy loss function as well as different state of the art methods for noisy and long tailed learning.

4.3.1 Overall accuracy

The overall accuracy for the datasets CIFAR-10 and CIFAR-100 are presented in the tables below.

Table 4.1: Overall accuracy on CIFAR-10 dataset

Imbalance ratio	**100**				**50**				**10**			
Noise rates	**0.2**	**0.3**	**0.4**	**0.5**	**0.2**	**0.3**	**0.4**	**0.5**	**0.2**	**0.3**	**0.4**	**0.5**
Baseline (CE)	60.76	54.1	40.95	36.62	63.85	55.21	49.88	44.37	77.43	72.3	63.28	59.31
DivideMix	66.61	63.99	62.68	58.14	75.29	73.59	71.42	67.7	88.13	87.95	86.47	87.29
HAR	56.82	50.29	42.93	36.53	65.17	61.11	57.14	54.23	80.95	77.38	68.92	63.12
RCAL*	75.81	72.76	69.78	65.05	-	-	-	-	86.46	84.58	83.48	80.80
RoLT+	72.59	70.07	67.84	64.62	77.26	76.29	76.07	75.87	87.93	88.16	87.59	87.28
SFA	79.31	77.0	75.31	72.5	85.16	83.25	82.49	78.65	92.49	91.51	90.99	89.36
OURs	80.21	78.13	76.2	72.84	86.56	84.58	83.16	79.36	93.6	92.66	90.76	90.36

*The result for RCAL[22] has been used directly from the paper.
From 4.1 We see that our method based on Instant centroid estimation by median of means using samples with predicted score greater than τ achieves superior performance in CIFAR-10 dataset. It beats the benchmark methods by about 1%. This improvement in the result might be attributed to the fact that for non normal datasets MoM which is a sub Gaussian estimator[26] performs better than the sample mean method if the data is noisy or long tailed. This is an empirical observation and its theoritical analysis based on the distribution of the samples in the fature space remains to be done.

Table 4.2: Overall accuracy on CIFAR-100 dataset

Imbalance ratio	100				50				10			
Noise rates	**0.2**	**0.3**	**0.4**	**0.5**	**0.2**	**0.3**	**0.4**	**0.5**	**0.2**	**0.3**	**0.4**	**0.5**
Baseline (CE)	30.17	26.81	19.53	15.24	33.1	24.61	22.87	17.15	46.25	41.33	34.49	30.17
DivideMix	41.91	38.43	33.93	30.57	49.25	45.1	41.37	35.78	62.54	58	53.57	49.61
HAR	27.09	22.23	18.45	13.94	29.91	25.48	21.64	17.32	45.2	39.57	29.44	27.31
RCAL*	39.85	36.57	33.36	30.26	-	-	-	-	54.85	51.66	48.91	44.36
RoLT+	45.02	41.63	38.27	34.57	51.39	47.62	43.04	39.01	64.08	60.26	56.61	52.11
SFA	47.14	45.15	40.59	36.4	51.33	49.21	45.42	40.94	64.88	61.59	58.34	54.42
OURs	47.79	45.37	40.16	35.76	52.01	49.92	46.04	41.42	65.94	62.58	59.32	55.5

*The result for RCAL has been used directly from the paper.

In 4.2 for the CIFAR-100 dataset the performance gain deteriorates as the imbalanced ratio and noise ratio goes up. Since, MoM is a little skewed for less number of samples so empirical evidence suggests that might be the reason why the performance deteriorates. Although the overall accuracy are still almost as good as the state of the art methods in most of the test cases.

4.3.2 Class wise results

We further analyse the class wise results for majority classes and minority classes as detailed below. It is important to analyse the class wise results to better understand the specifics of the model. While learning with imbalanced data is is important to gain acceptable generalizations from the minority classes too, otherwise it is as bad if not worse as learning without the minority class samples at all.

Table 4.3: Majority class accuracy on CIFAR-10 dataset

Imbalance ratio	100				50				10			
Noise rates	**0.2**	**0.3**	**0.4**	**0.5**	**0.2**	**0.3**	**0.4**	**0.5**	**0.2**	**0.3**	**0.4**	**0.5**
Baseline (CE)	94.51	92.34	86.75	80.40	90.81	83.03	77.76	71.22	94.25	90.36	87.23	85.37
DivideMix	74.64	75.29	80.56	76.83	85.61	84.58	80.70	78.14	92.31	92.62	95.21	93.46
RoLT+	87.58	83.57	82.14	75.31	86.31	82.56	83.52	86.31	92.14	93.10	95.35	95.64
SFA	92.40	90.21	90.12	89.97	96.80	96.50	94.70	91.15	97.55	98.05	96.10	94.25
OURs	96.50	92.50	91.57	91.05	96.20	95.95	94.75	93.30	97.10	97.50	96.80	96.95

Table 4.4: Minority class accuracy on CIFAR-10 dataset

Imbalance ratio	100				50				10			
Noise rates	**0.2**	**0.3**	**0.4**	**0.5**	**0.2**	**0.3**	**0.4**	**0.5**	**0.2**	**0.3**	**0.4**	**0.5**
Baseline (CE)	46.41	38.59	22.96	14.03	50.34	47.27	31.83	27.7	51.74	44.61	48.56	44.18
DivideMix	57.26	55.91	54.61	50.1	71.08	67.54	66.16	63.35	84.14	84.03	82.58	83.97
RoLT+	61.09	62.21	63.53	57.29	76.71	75.38	74.27	74.32	86.97	84.21	83.94	84.24
SFA	70.56	69.39	69.11	65.66	79.9	79.07	77.03	75.97	92.77	91.93	92.73	91.47
OURs	79.07	69.1	70.54	68.1	80.57	80.8	78.1	74.03	93.43	91.43	91.9	92.37

On experiments in CIFAR-10 dataset as shown in 4.3 and 4.4, We observe that the results for both majority and minority classes the accuracy are better than most of the state of the art methods. Specifically the majority class accuracies for imbalanced ratio 100 are significantly better than the comparison methods. Still while comparing with SFA we could not find any clear trend to suggest that our method is performing better or worse in one particular class or the other.

Table 4.5: Majority class accuracy on CIFAR-100 dataset

Imbalance ratio	**100**				**50**				**10**			
Noise rates	**0.2**	**0.3**	**0.4**	**0.5**	**0.2**	**0.3**	**0.4**	**0.5**	**0.2**	**0.3**	**0.4**	**0.5**
Baseline (CE)	41.56	39.61	36.52	35.14	40.59	33.21	31.65	29.48	52.74	50.74	43.25	34.51
DivideMix	50.92	44.63	40.18	40.21	56.47	48.02	48.92	42.42	71.65	67.52	60.43	51.44
RoLT+	55.73	57.61	52.71	47.27	61.76	59.24	56.72	56.37	65.28	71.65	65.73	56.9
SFA	65.28	61.69	59.44	55.17	67.32	62.7	60.27	54.76	68.4	65.04	62.87	58.76
OURs	65.03	61.64	58.71	54.94	65.95	63.6	61.37	55.59	70.01	65.53	62.63	59.46

Table 4.6: Minority class accuracy on CIFAR-100 dataset

Imbalance ratio	**100**				**50**			
Noise rates	**0.2**	**0.3**	**0.4**	**0.5**	**0.2**	**0.3**	**0.4**	**0.5**
Baseline (CE)	18.16	14.23	, 9.2	7.82	22.68	16.62	15.33	12.31
DivideMix	19.36	20.32	17.16	16.61	27.04	23.8	19.03	22.3
RoLT+	23.15	21.08	16.38	19.37	30.83	26.39	20.78	17.94
SFA	23.91	22.78	18.94	15.82	31.26	28.47	25.82	19.63
OURs	22.97	20.59	16.45	13.68	30.26	25.11	23.76	18.79

In CIFAR-100 we observe from 4.5 and 4.6 that for the minority class a clear trend that our method perform worse than the SFA. This empirically further strengthens our claim that due to low number of samples in such classes the MoM estimation works sub optimally.
We don't include the results for the medium classes in the paper as the don't provide any observable trends or meaningful insights.

4.3.3 Ablation Study

We experiment with the hyperparameters k (number of blocks for MoM) and n (number of times MoM is performed and taken average upon) to see its effect on the experiment results. We experiment this in CIFAR-00 dataset under two settings viz. Imbalanced ratio 50, noise ration 0.4 and imbalanced ration 10, noise ratio 0.5.

Table 4.7: Ablation study on two setup

ImbFactor_NoiseRatio	**50_0.4**				**100_0.5**			
NumBlocks	**1**	**5**	**10**	**20**	**1**	**5**	**10**	**20**
3	45.42	45.4	45.97	45.71	54.42	54.78	54.68	54.92
5	45.6	45.42	45.89	46.03	55.39	55.26	55.33	55.5
10	45.41	45.23	45.69	46.04	54.83	54.92	55.3	55.2
20	45.02	45.21	45.32	45.64	54.51	55.17	55.02	54.98

Theoretically, as the number of samplings increase the mean estimation will become more and more robust. This is because in the median of means method the sample is divided into k blocks of similar size, their means calculated and then the median of those means are taken as estimate for the population mean. If in this random sampling process the shuffled groups exhibits any trend then the MoM estimator will be biased. To address this and design a robust estimator the MoM can be performed for all the possible permutations for the k groups and their average taken as the robust MoM estimate. Since that is practically impossible so instead taking n number of samples solves the problem. So it is evident as n increases the estimate should become better and better.

From 4.7 we observe that there is a little increase in the accuracy as we increase the number of samplings. We observe the best results for number of blocks 5 and 10. We think this is due to the fact that even in the smaller classes there are enough data to perform well on this many blocks but increasing the number of blocks further results in lesser number of samples in each block and that affects the efficiency of the MoM estimator.

Chapter 5

Conclusions and Future Work

In our work, we have proposed an intuitive approach, giving a strategy that is easy to understand and apply. Our proposed method can be combined in other existing works that use a similar approach. Our method clearly outperforms the baselines in the CIFAR-10 dataset and outperforms the baselines in low to medium imbalanced setup on CIFAR-100 dataset.

We would try to examine whether our proposed method for calculating mean estimate has sound theoretical backing. We claimed empirically that the results in the datasets with very few samples in many classes affects the estimation but it remains to be proved in its entirety. If that is indeed the case a line of work can be to use MoM for the classes where there is an acceptable amount of data and use normal mean or some other estimator that is robust to data scarcity for the other classes. There are statistically better sub Gaussian mean estimators than MoM. Their implementation can be studied and experimented with in our setup.

After doing the comparative study for the existing approaches we have identified a few lines of work for further enhancement of the model. (i) To use a different and more robust loss function for the balanced classifier such as focal loss. This might improve the representation of the features. (ii) Instead of warm up with CE and BCE losses we can use a learning method that does not rely on the labels for learning like contrastive learning, this might help in learning with high level of noise. (iii) To examine the effectiveness of the Gaussian Mixture model in identifying the clean samples and investigate applicability of alternative approaches in our setup. (iv) The semi supervised learning uses MixMatch. There has been some advances in the area of SSL like FixMatch, ReMixMatch, UDA, FocalMatch etc. We intend to use those methods with our setup and check their effectiveness.

Bibliography

[1] K. Cao, Y. Chen, J. Lu, N. Arechiga, A. Gaidon, and T. Ma, "Heteroskedastic and imbalanced deep learning with adaptive regularization," 2021.

[2] T. Wei, J.-X. Shi, Y.-F. Li, and M.-L. Zhang, "Prototypical classifier for robust class-imbalanced learning," 2021.

[3] T. Wei, J.-X. Shi, W.-W. Tu, and Y.-F. Li, "Robust long-tailed learning under label noise," 2021.

[4] J. Shu, Q. Xie, L. Yi, Q. Zhao, S. Zhou, Z. Xu, and D. Meng, "Meta-weight-net: Learning an explicit mapping for sample weighting," 2019.

[5] B. Kang, S. Xie, M. Rohrbach, Z. Yan, A. Gordo, J. Feng, and Y. Kalantidis, "Decoupling representation and classifier for long-tailed recognition," 2020.

[6] A. K. Menon, S. Jayasumana, A. S. Rawat, H. Jain, A. Veit, and S. Kumar, "Long-tail learning via logit adjustment," 2021.

[7] H. Lee, S. Shin, and H. Kim, "Abc: Auxiliary balanced classifier for class-imbalanced semi-supervised learning," 2021.

[8] Y. Cui, M. Jia, T.-Y. Lin, Y. Song, and S. Belongie, "Class-balanced loss based on effective number of samples," 2019.

[9] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, pp. 2980–2988, 2017.

[10] K. Cao, C. Wei, A. Gaidon, N. Arechiga, and T. Ma, "Learning imbalanced datasets with label-distribution-aware margin loss," 2019.

[11] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," *arXiv preprint arXiv:1710.09412*, 2017.

[12] H. Chou, S. Chang, J. Pan, W. Wei, and D. Juan, "Remix: Rebalanced mixup," *CoRR*, vol. abs/2007.03943, 2020.

[13] A. Galdran, G. Carneiro, and M. A. González Ballester, "Balanced-mixup for highly imbalanced medical image classification," in *Medical Image Computing and Computer Assisted Intervention–MICCAI 2021: 24th International Conference, Strasbourg, France, September 27–October 1, 2021, Proceedings, Part V 24*, pp. 323–333, Springer, 2021.

[14] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo, "Cutmix: Regularization strategy to train strong classifiers with localizable features," *CoRR*, vol. abs/1905.04899, 2019.

[15] G. Patrini, A. Rozza, A. Menon, R. Nock, and L. Qu, "Making deep neural networks robust to label noise: a loss correction approach," 2017.

[16] D. Cheng, T. Liu, Y. Ning, N. Wang, B. Han, G. Niu, X. Gao, and M. Sugiyama, "Instance-dependent label-noise learning with manifold-regularized transition matrix estimation," 2022.

[17] Z. Zhang and M. R. Sabuncu, "Generalized cross entropy loss for training deep neural networks with noisy labels," 2018.

[18] S. Liu, J. Niles-Weed, N. Razavian, and C. Fernandez-Granda, "Early-learning regularization prevents memorization of noisy labels," 2020.

[19] B. Han, Q. Yao, X. Yu, G. Niu, M. Xu, W. Hu, I. Tsang, and M. Sugiyama, "Co-teaching: Robust training of deep neural networks with extremely noisy labels," 2018.

[20] J. Li, R. Socher, and S. C. H. Hoi, "Dividemix: Learning with noisy labels as semi-supervised learning," 2020.

[21] H. Permuter, J. Francos, and I. Jermyn, "A study of gaussian mixture models of color and texture features for image classification and segmentation," *Pattern Recognition*, vol. 39, pp. 695–706, 04 2006.

[22] M. Zhang, X. Zhao, J. Yao, C. Yuan, and W. Huang, "When noisy labels meet long tail dilemmas: A representation calibration method," 2023.

[23] H.-T. Li, T. Wei, H. Yang, K. Hu, C. Peng, L.-B. Sun, X.-L. Cai, and M.-L. Zhang, "Stochastic feature averaging for learning with long-tailed noisy labels," pp. 3902–3910, 08 2023.

[24] D. Berthelot, N. Carlini, I. Goodfellow, N. Papernot, A. Oliver, and C. Raffel, "Mixmatch: A holistic approach to semi-supervised learning," 2019.

[25] S. Minsker, "Efficient median of means estimator," 2023.

[26] L. Devroye, M. Lerasle, G. Lugosi, and R. I. Oliveira, "Sub-gaussian mean estimators," 2015.

[27] E. Arazo, D. Ortego, P. Albert, N. E. O'Connor, and K. McGuinness, "Unsupervised label noise modeling and loss correction," 2019.

[28] D. Arpit, S. Jastrzebski, N. Ballas, D. Krueger, E. Bengio, M. S. Kanwal, T. Maharaj, A. Fischer, A. Courville, Y. Bengio, and S. Lacoste-Julien, "A closer look at memorization in deep networks," 2017.

[29] J. Cui, Z. Zhong, S. Liu, B. Yu, and J. Jia, "Parametric contrastive learning," 2021.

[30] A. Ghosh, H. Kumar, and P. S. Sastry, "Robust loss functions under label noise for deep neural networks," 2017.

[31] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," 2016.

[32] D. Hendrycks, M. Mazeika, D. Wilson, and K. Gimpel, "Using trusted data to train deep networks on labels corrupted by severe noise," 2019.

[33] M. A. Jamal, M. Brown, M.-H. Yang, L. Wang, and B. Gong, "Rethinking class-balanced methods for long-tailed visual recognition from a domain adaptation perspective," 2020.

[34] L. Jiang, Z. Zhou, T. Leung, L.-J. Li, and L. Fei-Fei, "Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels," 2018.

[35] S. Jiang, J. Li, Y. Wang, B. Huang, Z. Zhang, and T. Xu, "Delving into sample loss curve to embrace noisy and imbalanced data," 2021.

[36] Z. Liu, Z. Miao, X. Zhan, J. Wang, B. Gong, and S. X. Yu, "Large-scale long-tailed recognition in an open world," 2019.

[37] W. Maddox, T. Garipov, P. Izmailov, D. Vetrov, and A. G. Wilson, "A simple baseline for bayesian uncertainty in deep learning," 2019.

[38] M. Ren, W. Zeng, B. Yang, and R. Urtasun, "Learning to reweight examples for robust deep learning," 2019.

[39] J. Ren, C. Yu, S. Sheng, X. Ma, H. Zhao, S. Yi, and H. Li, "Balanced meta-softmax for long-tailed visual recognition," 2020.

[40] L. Shen, Z. Lin, and Q. Huang, "Relay backpropagation for effective learning of deep convolutional neural networks," 2016.

[41] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," 2016.

[42] K. Tang, J. Huang, and H. Zhang, "Long-tailed classification by keeping the good and removing the bad momentum causal effect," 2021.

[43] T. Wei and Y.-F. Li, "Does tail label help for large-scale multi-label learning?," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 7, pp. 2315–2324, 2020.

[44] Z.-F. Wu, T. Wei, J. Jiang, C. Mao, M. Tang, and Y.-F. Li, "Ngc: A unified framework for learning with open-world noisy data," 2021.

[45] X. Xia, T. Liu, B. Han, M. Gong, J. Yu, G. Niu, and M. Sugiyama, "Sample selection with uncertainty of losses for learning with noisy labels," 2021.

[46] L. Xiang, G. Ding, and J. Han, "Learning from multiple experts: Self-paced knowledge distillation for long-tailed classification," 2020.

[47] X. Yi, K. Tang, X.-S. Hua, J.-H. Lim, and H. Zhang, "Identifying hard noise in long-tailed sample distribution," 2023.

[48] B. Zhou, Q. Cui, X.-S. Wei, and Z.-M. Chen, "Bbn: Bilateral-branch network with cumulative learning for long-tailed visual recognition," 2020.

[49] X. Zhou, X. Liu, D. Zhai, J. Jiang, X. Gao, and X. Ji, "Prototype-anchored learning for learning with imperfect annotations," 2022.

[50] Z.-H. Zhou, "Open-environment machine learning," 2022.