THE UNIVERSITY OF
SYDNEY

COMP5328 Assignment 2

Author (SID):

Jiayuan Ma (SID: 520370197)

Jing Luo (SID: 540033724)

Kaiyuan Xu (SID: 500521681)

Zechen Zhu (SID: 540323234)

November 5, 2024

Contents

Abstract

This report focuses on how to construct an anti-noise classification model under the influence of annotation noise. To achieve this goal, two classification algorithms and a transition matrix estimation method are proposed to cope with random annotation noise in the training and validation datasets. In terms of method, firstly, based on the CIFAR and FashionMNIST datasets, the baseline model was established by Convolutional Neural Network (CNN), and different loss functions such as cross-entropy loss, reweighted loss and generalized cross-entropy loss were applied, and the transition matrix was estimated by combining the Expectation Maximization (EM) algorithm. In the experimental part, Top-1 accuracy is used as the main performance metric to evaluate the performance of the model under different noise levels. The structure of the report includes introduction, related studies, methods, experiments, and analysis of results. Finally, the effectiveness and applicability of the anti-noise method are summarized, which provides a reference for improving the robustness of the model in the annotation noise environment in the future.

1 Introduction

Label noise is a key problem in machine learning, and mislabeled data can seriously affect model performance. Label noise is caused by errors in the annotation process, such as automatic annotation errors and human errors [3]. This issue is particularly important in domains that require high accuracy, such as banking, autonomous vehicles, and medical diagnostics, where inaccurate predictions of label noise can have expensive or even harmful effects. Thus, creating algorithms that can withstand label noise is essential for reliable, practical machine learning applications.

The objective of this paper is to provide two classification algorithms and a transition matrix estimator that is resistant to label noise. The test set will be clean, while the training set and validation set will use three data sets with random label noise. The transition matrix of the first two data sets is given, which can be directly used to construct a classifier that is resistant to label noise. For the third dataset, we will use a transition matrix estimator on this noise structure, where the transition matrix is unknown. The classifier will be trained using an estimation matrix to improve its performance on clean test data. By evaluating the recovery ability of each algorithm on different data sets, an effective method is found to reduce the influence of label noise on classification tasks.

Our group plans to build a model using CNN as the baseline model and then use the expectation maximization algorithm to estimate the transition matrix T for the CIFAR dataset. We then use two different algorithms, the cross-entropy loss function and the reweighted loss function, to build the noise robust classifier. Finally, the top-1 accuracy is used as an evaluation index to evaluate the noise robustness of the classifier.

2 Related Work

In supervised learning, label noise refers to the phenomenon that labels in training data are incorrectly labeled. This is common in practical applications and can be caused by human error, sensor failure, or errors in automatic labeling systems. Label noise will reduce the performance of the model and reduce the generalization ability of the model. Therefore, studying how to build a robust classifier in the presence of label noise has become an important topic in the field of machine learning.

Researchers have proposed various approaches to solve the label noise problem. Among them, the model based on noise transfer matrix has been widely studied. The noise transfer matrix describes the probability distribution of real labels being mistaken for other labels. By estimating this matrix, the loss function can be modified during training, thereby mitigating the effects of noise. Patrini et al. proposed an unbiased method to estimate the noise transfer matrix, and modified the loss function on this basis, and achieved good results [8].

Another class of methods focuses on designing loss functions that are robust to noise. Some researchers proposed the symmetric cross entropy loss function, which combined the traditional cross entropy loss and inverse cross entropy loss to enhance the robustness of the model to noisy labels [9]. In addition, strategies based on small loss sample selection have also been proposed, such as the Co-teaching method proposed, which avoids the interference of noise samples by training two networks and selecting small loss samples from each other for training [4].

Recently, the memorization effect of deep learning models has been used to deal with label noise. Arpit et al. found that deep networks tend to learn simple and clean samples at the beginning of training, and then gradually memorize noise samples [2]. Using this property, an early stopping strategy is adopted to prevent the model from overfitting noisy labels, thus improving the generalization performance of the model.

In addition, semi-supervised learning methods have also been used to deal with the label noise problem. Some researchers proposed DivideMix method, which combines semi-supervised learning with mixture model to enhance the robustness of the model by generating pseudo-labels for unlabeled data [5]. The proposed method performs well when dealing with high noise rate datasets.

In summary, the processing methods for label noise mainly include correction based on a noise transition matrix, design of robust loss function, use of memory effect of deep model, and combination of semi-supervised learning. These methods show their advantages in different application scenarios, but they also have certain limitations, such as the assumption of noise distribution and computational complexity. Therefore, choosing an appropriate method for a specific task and combining multiple strategies may be an effective way to cope with the label noise problem.

3 Methods

Our group used three datasets, namely CIFAR, FashionMNIST0.3, and FashionMNIST0.6. The computer vision dataset CIFAR contains 24000 color pictures

in 4 categories, each measuring 32 × 32 × 3 (width × height × RGB channels).

Furthermore, FashionMNIST is a dataset of clothes photographs that takes the place of the MNIST handwritten digit dataset. Its goal is to provide image classification models more difficult images. A variation of the original FashionMNIST dataset with 30% label noise is called FashionMNIST0.3. FashionMNIST single-channel grayscale pictures are 28 x 28 in size.

Additionally, FashionMNIST0.6 is comparable to FashionMNIST0.3, but it has a greater percentage of label noise—up to 60% of the labels are jumbled at random. Approximately 60% of sample labels in both training and validation data do not correspond to the true category. The robustness of the classification model is further challenged in this dataset due to its larger amount of noise than in FashionMNIST0.3. In addition to having 4 categories, this dataset comprises 28 × 28 images.

3.1 Pre-processing: Normalization

To increase the effectiveness and stability of model training, our team normalizes the picture data by scaling it to a predetermined range, often between 0 and 1. Since each picture pixel value, particularly in 8-bit photos, often ranges from 0 to 255, the image data is divided by 255. All pixel values are scaled to 0 to 1 by dividing these values by 255. We chose this approach because a wide range of feature values during machine learning model training might result in a sluggish convergence of the optimization process. The model's convergence can be accelerated by scaling the data such that the features are on the same order of magnitude.

Additionally, it can enhance model performance. Normalization can lower the data's deviation, particularly for models that are sensitive to distance. We employ the following Min-Max normalization formula:

$X_{\text{normalized}} = \frac{X - X_{\text{min}}}{X_{\text{max}} - X_{\text{min}}}$ [1]

Where:

X normalized is the normalized data,

X is the original data (e.g., pixel value),

X max is the maximum value in the data (here 255),

X min is the minimum value in the data (here 0).

3.2 Method 1: Cross-entropy Loss function

The cross-entropy loss function is a measure of the difference between 2 probability distributions. The cross-entropy loss measures the difference between the true label distribution and the probability distribution of the model's output. The smaller the cross entropy, the closer the two distributions are, indicating that the true label is closer to the probability distribution of the expected class. The mathematical formula is as follows:

$L_{\text{CE}} = -\sum_i y_i \log(\hat{y}_i)$ [10]

where:

$y_i$is the true label,

$\hat{y}_i$ is the probability distribution predicted by the model,

n is the number of classification categories.

The objective of the cross entropy loss is to reduce this value in order to get the model's projected probability output as near to the distribution of the real label as feasible.

3.3 Method 2: Reweight Loss Function

The Reweighted Loss Function modifies each sample's contribution to the overall loss according to predicted noise probabilities to handle label noise in classification tasks. This approach supports the development of robust classifiers that are less susceptible to noisy labels by prioritizing clean data and decreasing the weight on noisy samples. While categorical cross-entropy is the foundation of the reweighted loss function, each sample is given importance weights. The function may be shown like this:

$L_{\text{Reweighted}} = -\sum_i w_i \cdot y_{\text{true}_i} \log(\hat{y}_{\text{pred}_i})$ [7]

Where:

$L_{\text{Reweighted}}$represents the reweighted loss.

$w_i$is the weight for each instance.

y_{true_i}represents the true label.

$\hat{y}_{\text{pred}_i}$is the predicted probability after applying softmax.

3.4 Method 3:Generalized Cross Entropy Loss Function

A robust loss function, called Generalized cross-entropy (GCE) loss function, is created to reduce the harmful effects of noisy labels and improve the reliability of the model in the presence of mislabeled data. The proposed method combines elements of the traditional Mean Absolute Error (MAE) and cross-entropy loss to provide a more flexible cost function. When dealing with noisy or mislabeled data, standard loss functions like cross-entropy may not be ideal because they have a tendency to overfit and generalize poorly due to an excessive focus on the errors associated with mislabeled samples. To alleviate these problems, the GCE loss function is introduced to balance the influence of each sample. The GCE loss function is formulated as:

$L_{\text{GCE}}(y_{\text{true}}, \hat{y}_{\text{pred}}) = \frac{1-(\hat{y}_{\text{pred}})^q}{q}$ [9]

Where:

$L_{\text{GCE}}$is the generalized cross-entropy loss.

y_{true_i} is the true label.

$\hat{y}_{\text{pred}_i}$is the predicted probability.

q is a parameter that controls the robustness of the loss function to label noise.

3.5 Method 4:Expectation-Maximization

The Expectation-Maximization algorithm is a widely used method for estimating parameters in models with latent variables or hidden structures. Here, we use EM to estimate a transition matrix T that represents the probabilities of noisy class transitions due to label noise. The algorithm alternates between two main

steps: the Expectation step, where it predicts latent variables, and the Maximization step, where it updates the model parameters, gradually refining T.

Given the current estimate of T, calculate the probability of each sample's label under noisy conditions:

$$\text{noisy\_probs}_{i,j} = \frac{\sum_k \text{pred\_probs}_{i,k} \cdot T[k,j]}{\sum_m \sum_k \text{pred\_probs}_{i,k} \cdot T[k,m] + \epsilon}$$

[11]
where pred_probs represents the predicted class probabilities from the model.

To update each element of T, compute the average noisy probability for class J samples:

$$T[i,j] = \frac{\sum_l \text{noisy\_probs}_{l,i} \cdot 1(y_l = j)}{\sum_l 1(y_l = j)} + \text{noise}$$

[6]
where $1(y_l = j)$ is an indicator function that selects samples of class j.

4 Experiment

4.1 Data Preparation

In the data preparation part of this experiment, we loaded, analyzed, and visualized multiple datasets to prepare for subsequent model training and evaluation.

Firstly, We loaded three datasets CIFAR, FashionMNIST0.3 and FashionMNIST0.6, which contain training and testing data and their labels, respectively. The CIFAR dataset consists of 20,000 samples for training and 4,000 samples for testing. Each image is 32x32 pixels in size and is a 3-channel color image (RGB). The FashionMNIST0.3 and FashionMNIST0.6 datasets contain 24,000 training samples and 4,000 test samples respectively. The size of each image is 28x28 pixels, single channel grayscale image. All three datasets contain 4-class labels covering different categories of images or clothing styles.

After that, we performed a visualization of each dataset in figure 1, figure 2 and figure 3, randomly showing part of the samples and their labels, so as to understand the distribution and characteristics of the data more intuitively.

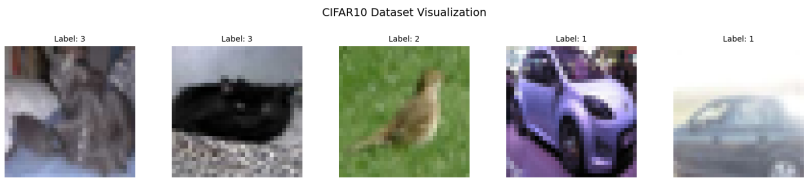

Figure 1: Five Samples of CIFAR Dataset

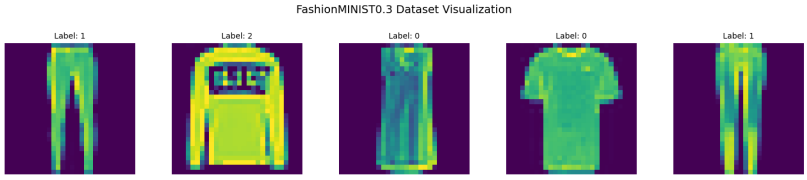

Figure 2: Five Samples of FashionMNIST0.3 Dataset

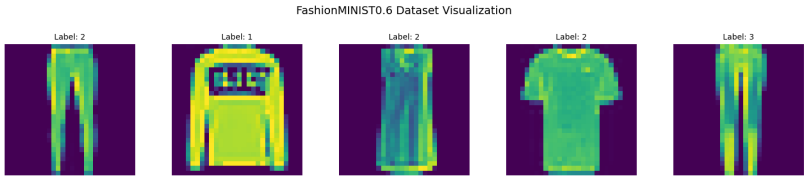

Figure 3: Five Samples of FashionMNIST0.6 Dataset

Finally, after we split the dataset into training and validation sets, we normalize all image pixel values by dividing the pixel values by 255 and scaling the value range to between [0,1]. This processing step can improve the efficiency of model training, help the model converge faster, and provide standardized data input for the subsequent training process.

4.2 CNN Model

In the CNN model building part of this experiment, we designed and trained a convolutional neural network (CNN) model for image classification task. First, we define a CNN model constructor that takes the shape of the input image and the number of categories as parameters. The model structure consists of two convolutional layers, each of which is followed by a max-pooling layer using 32 and 64 convolution kernels with a convolution kernel size of (3,3) and a pooling window size of (2,2), respectively. The ReLU activation function is used after each convolutional layer, and the features are flattened by a Flatten layer after pooling. To prevent overfitting, a Dropout layer is added to the model and the drop rate is set to 50%. The flatten layer is followed by a fully connected layer of 128 units with ReLU activation function, and the final output layer sets the number of units according to the number of categories and uses softmax activation function for multi-class classification.

The model was compiled using Adam optimizer and the learning rate was set to 0.01. The loss function is ”sparse categorical cross-entropy” to fit the multi-class label format, and the evaluation metric is accuracy. After completing the model construction and compilation, we trained the model on different datasets, using 10 epochs and 64 batch sizes.

When training the CNN model on the CIFAR dataset, the accuracy of the model on the training set is gradually increased from 32.45% to 55.42%, and the accuracy on the validation set is also gradually increased to 53.73%. The training performance of the FashionMNIST0.3 dataset is slightly better, the accuracy is improved from 54% to 64.68%, and the accuracy of the validation set is improved from 63.58% to 66.37%. On the FashionMNIST0.6 dataset, the

accuracy of the initial training is 28.11%, and it reaches 32.73% after training. The accuracy on the validation set decreases slightly from 36.19% to 34.04%. It can be seen from the results that there are differences in the convergence speed and accuracy of different datasets in the training process, but the overall model shows a good training effect on all datasets.

4.3 Expectation-Maximization (EM) Algorithm

The Expectation-Maximization (EM) algorithm is used to estimate the transition matrix of the data set to simulate the generation process of label noise. Specifically, we define a function 'estimate transition matrix' that takes the model, input data, and label data as parameters and iterates to update the transition matrix. First, we initialize a random transition matrix and set the number of iterations to 15. In each iteration, the predicted probabilities of the model on the data are first calculated, and these probability values are softened, and then the current transition matrix is used to update the noise probabilities.

In the E-step, we calculate the noise probability using the predicted probability of the model and the current transition matrix. Then, in the m-th step, the columns of the transition matrix are updated according to the data labels to minimize the noise probability and the error of the actual label. To prevent numerical instability, we add a small epsilon value during matrix normalization to ensure that each column sums to one. After several iterations, a stable transition matrix is finally obtained.

On the CIFAR dataset, we generated the transition matrix using this algorithm. In addition, to verify the algorithm's accuracy, we compare with the known transition matrices of FashionMNIST0.3 and FashionMNIST0.6 datasets. By calculating the error matrix between the estimated transition matrix and the known transition matrix, it can be seen that the average error on the FashionMNIST0.3 dataset is 0.246432, while the average error on the FashionMNIST0.6 dataset is 0.074762. Using these results, we can evaluate the EM algorithm's estimation effect on different datasets and demonstrate its effectiveness in estimating the transition matrix.

4.4 Loss Function

2 different loss functions, reweighted loss function and Generalized Cross Entropy (GCE) loss function, were designed and implemented to deal with the label noise problem.

Firstly, we define the Reweighted Loss Function, where the 'importance weighted loss' function adjusts the prediction probability by the input transition matrix T. In this loss function, the predicted probability of the model output is subjected to a softmax transformation and then adjusted to a probability distribution with noise by the transition matrix T. The true labels are converted to one-hot encoding, and the weighted categorical cross-entropy with the adjusted prediction probability is calculated to obtain the final loss. Using this weighting mechanism can effectively counteract the effect of label noise and enable the model to classify the samples more accurately during training. In the training

phase, we define the ”training Importance weighted model” function to train the model using the reweighted loss function and record the accuracy of the model.

Secondly, a generalized cross-entropy loss function (GCE loss function) was implemented to further reduce the impact of noisy labels. The GCE loss function adjusts the noise robustness through the parameter q. Specifically, the ‘generalized cross-entropy’ function computes the softmax probability of the model output and computes the generalized cross-entropy loss by extracting the corresponding predicted probability based on the one-hot encoding of the true label. For q close to 0, the GCE approaches the conventional cross-entropy, while for increasing q values, the loss function becomes more robust to noise. We use the ‘train gce loss model’ function to apply the GCE loss function during training to ensure that the model maintains high classification accuracy in noisy data.

By designing and applying these two different loss functions, the influence of label noise on model training can be effectively dealt with, and more robust performance can be obtained in model training.

4.5 Model Evaluation

In the Model Evaluation section, we evaluate the model performance on CIFAR and FashionMNIST datasets. We focus on three different Loss functions: Cross Entropy, Importance Weighted Loss, and Generalized Cross Entropy (GCE). During the evaluation, each model and dataset combination was run 10 times to see the stability and accuracy performance of the model across multiple runs.

Firstly, for evaluating the CIFAR dataset, the results show that the cross-entropy loss model has the highest average Top-1 accuracy, reaching about 63%, as shown in figure 4. This shows that the cross-entropy loss performs optimally in the standard noiseless scenario. However, after introducing the importance of weighted loss, the average accuracy of the model drops significantly, only about 38%, indicating that the weighted loss does not work well for this dataset. The GCE model is somewhere in between, with an average accuracy of around 54%. The contrast of these accuracies can be seen in the heat maps of figure 9.

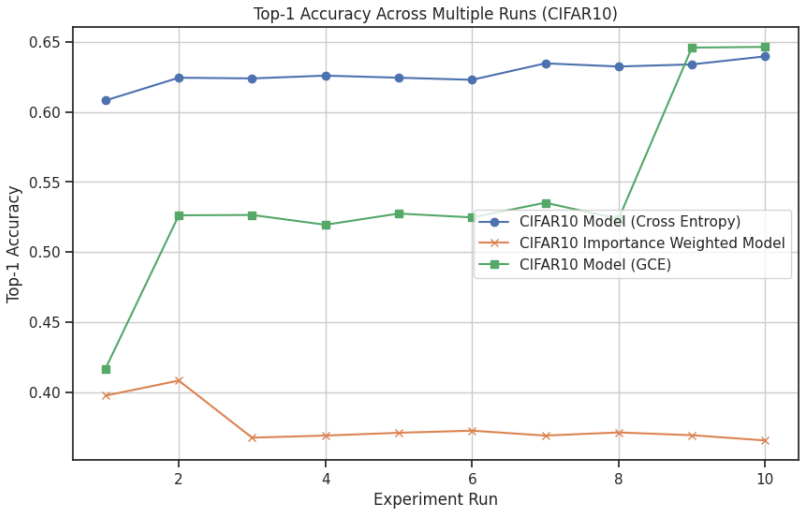

Figure 4: Top-1 Accuracy Across Multiple Runs in CIFAR Dataset

For the FashionMNIST0.3 dataset, the model performs differently. The average Top-1 accuracy of the cross-entropy model is about 67%, while the accuracy of the importance weighted and GCE models both reaches about 68%, as shown in figure 5and figure 9. This shows that on the FashionMNIST0.3 dataset, the GCE and importance weighting strategies can slightly improve the anti-noise ability and accuracy of the model.

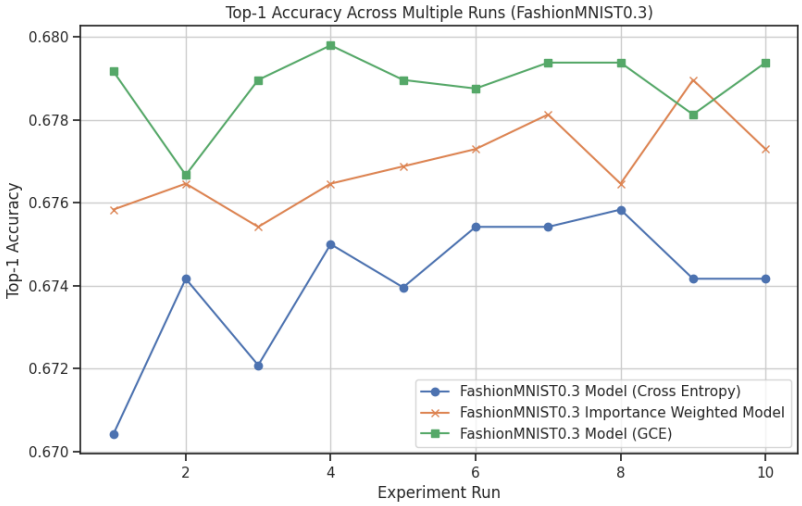

Figure 5: Top-1 Accuracy Across Multiple Runs in FashionMNIST0.3 Dataset

On the FashionMNIST0.6 dataset, the overall accuracy of all models is low, with the cross-entropy model achieving an average accuracy of 37%, while the importance weighted sum GCE model achieves an accuracy of about 38%, as shown in figure 6 and figure 9. It can be seen that on the more challenging FashionMNIST0.6 dataset, the GCE and weighted loss models perform slightly better than the cross-entropy loss model, but the improvement is more limited.

Figure 6: Top-1 Accuracy Across Multiple Runs in FashionMNIST0.6 Dataset

Figure 7 shows the accuracy variation trend of the three models in 10 experiments on each data set. It can be seen that FashionMNIST0.3 performs

relatively well and stably in all three models. CIFAR can achieve a high level of accuracy on the cross-entropy and generalized cross-entropy models, but performs poorly on the importance weighted loss and does not achieve accuracy improvement with the increase of the number of iterations. For FashionMNIST0.6, the performance is relatively poor in all three models, and there is no obvious improvement in accuracy in the process of experimental iteration. In addition, figure 8 presents the average accuracy and its standard deviation for each combination of model and dataset, further illustrating the difference in model performance on different datasets.

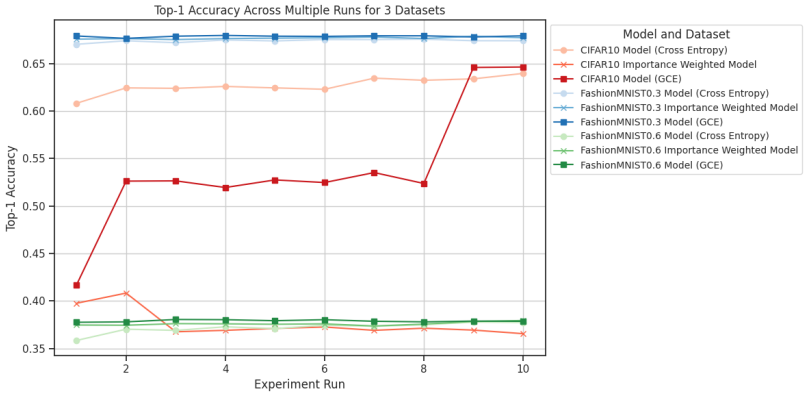

Figure 7: Top-1 Accuracy Across Multiple Runs in All 3 Dataset

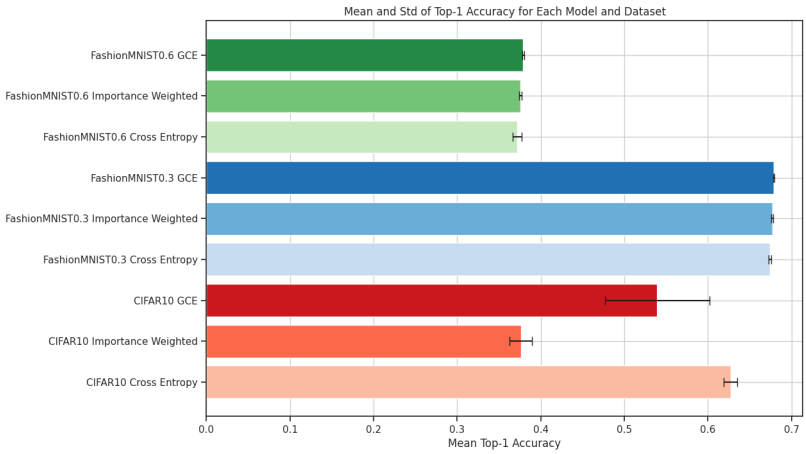

Figure 8: Mean and Std of Top-1 Accuracy for Each Model and Dataset

In summary, from the experimental results, it can be seen that the traditional cross-entropy loss is still the most effective for standard datasets (such as CIFAR). For the data sets with high noise (such as FashionMNIST0.3 and FashionMNIST0.6), the importance weighting and GCE loss can improve the accuracy and stability of the model to a certain extent. Future research and application can choose the appropriate loss function according to the characteristics of the data to improve the generalization performance and stability of the model.

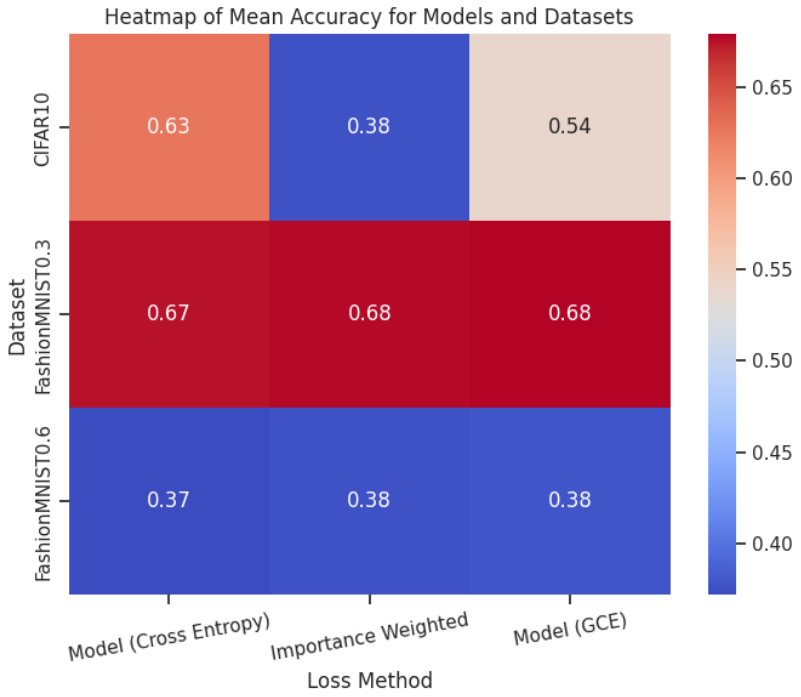

Figure 9: Heatmap of Mean Accuracy for Models and Datasets

5 Conclusion

As stated in our report, we find that different models and datasets perform differently:

When it comes to the CIFAR dataset, the cross-entropy model consistently outperforms the importance-weighted model and the generalized cross-entropy (GCE) model in terms of accuracy over several runs. At the end of the run, the GCE model notably improved its accuracy, indicating that anti-noise training would eventually be advantageous.

Based on the FashionMNIST0.3 data set, the GCE model outperforms the importance-weighted and cross-entropy models in terms of accuracy. As the simulation progresses, the importance-weighted model gradually closes the gap with the GCE model. The cross-entropy model consistently had the lowest accuracy, indicating that it might not be as effective as other approaches in handling label noise.

For the FashionMNIST0.6 data set, when the noise level is high, the GCE model continues to outperform other models and maintains a relatively stable accuracy. While the difference between the two models is less than that of the FashionMNIST0.3 dataset, the importance-weighted model performs better than the cross-entropy model, particularly in later runs.

In conclusion, for all datasets, the extended cross-entropy model demonstrates resistance to label noise, particularly in high-noise situations. Compared to the traditional cross-entropy model, which routinely performs badly, the importance-weighted model exhibits a little improvement and adjusts to noise better. These results imply that model performance in noisy situations can be enhanced by applying anti-noise techniques such as GCE. Future research

should examine more ways to improve noise resilience and examine how stable GCE is at various noise levels and kinds.

In the future, we can explore hybrid approaches, combining GCE with other robust training techniques to further improve the model's resilience to label noise. Second, future experiments can include more diverse datasets to verify the generalizability of the observed trends. More importantly, studying methods to optimize the computational efficiency of GCE, especially for large-scale datasets, is crucial for practical applications.

References

[1] Peshawa Jamal Muhammad Ali, Rezhna Hassan Faraj, Erbil Koya, Peshawa J Muhammad Ali, and Rezhna H Faraj. Data normalization and standardization: a technical report. *Mach Learn Tech Rep*, 1(1):1–6, 2014.

[2] Devansh Arpit, Stanislaw Jastrzebski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S. Kanwal, Tegan Maharaj, Asja Fischer, Aaron C. Courville, Yoshua Bengio, and Simon Lacoste-Julien. A closer look at memorization in deep networks. In *International Conference on Machine Learning*, 2017.

[3] Benoît Frénay and Michel Verleysen. Classification in the presence of label noise: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 25:845–869, 2014.

[4] Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor Wai-Hung Tsang, and Masashi Sugiyama. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *Neural Information Processing Systems*, 2018.

[5] Junnan Li, Richard Socher, and Steven C. H. Hoi. Dividemix: Learning with noisy labels as semi-supervised learning. *ArXiv*, abs/2002.07394, 2020.

[6] Todd K. Moon. The expectation-maximization algorithm. *IEEE Signal Process. Mag.*, 13:47–60, 1996.

[7] Yuta Mukobara, Yutaro Shigeto, and Masashi Shimbo. Rethinking loss functions for fact verification. *ArXiv*, abs/2403.08174, 2024.

[8] Giorgio Patrini, Alessandro Rozza, Aditya Krishna Menon, Richard Nock, and Lizhen Qu. Making deep neural networks robust to label noise: A loss correction approach. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2233–2241, 2016.

[9] Zhilu Zhang and Mert Rory Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. *ArXiv*, abs/1805.07836, 2018.

[10] Yangfan Zhou, Xin Wang, Mingchuan Zhang, Junlong Zhu, Ruijuan Zheng, and Qingtao Wu. Mpce: A maximum probability based cross entropy loss function for neural network classification. *IEEE Access*, 7:146331–146341, 2019.

Appdendix

A. Contribution

1. Jiayuan Ma (SID: 520370197)
Jiayuan Ma was mainly responsible for writing related work in the report firstly, where include reviews the main methods to deal with annotation noise, including the correction based on noise transition matrix, the design of anti-noise loss function, the use of memory effect of the model to avoid overfitting noise labels, and the generation of pseudo-labels through semi-supervised learning to improve the anti-noise ability of the model. Then, in the experiment part, he describes the flow of the whole experiment, the model and parameters used, and then interprets and analyzes the results. Finally, he completed the writing of the abstract section, summarizing the content of the whole report in concise words.

2. Jing Luo (SID: 540033724)
Jing Luo was responsible for the introduction, which included the purpose of the group's report and the methods used by the group to achieve it. Secondly, she was also responsible for the method, which included the explanation and principle of cross entropy loss function, reweight loss function, Generalized Cross Entropy Loss Function and expectation Maximization, as well as the corresponding implementation formulas. More importantly, she was also responsible for the conclusion. Based on the image results of the code running, the results of the three methods on three data sets were compared. Finally, she was responsible for modifying the report and doing the final arrangement.

3. Kaiyuan Xu (SID: 500521681)
Kaiyuan Xu was responsible for the later content writing of the code, focusing on Task 3.2. He used the EM algorithm to predict the transformation matrix T of the dataset CIFAR.npz. He also built three classifiers with Zechen. He added another loss function to Zechen's original two classifiers (Cross Entropy and Reweighted Loss Function) to build a classifier (Generalized Cross Entropy (GCE) Loss Function). In addition, he finally visualized the evaluation results of the three classifiers in three ways (line chart, bar chart, heat map) and the overall arrangement of the Jupyter book.

4. Zechen Zhu (SID: 540323234)
Zechen Zhu was responsible for the initial content writing of the code, primarily focusing on task 3.1, where he used the given transition matrices to build classifiers. Specifically, he utilized CNN as the baseline model, with the loss function set to cross-entropy. Additionally, another classifier was modified to use an importance-weighted loss.

B. Execution Environment

This study utilised Colab to execute the code and record experimental results, and elevant Python libraries, including *numpy, pandas, matplotlib, scikit-learn and softmax*, etc., were imported for data manipulation, visualisation, clustering, and metric calculation. The data file contains the CIFAR, FashionMNIST0.3 and FashionMNIST0.6 datasets.

Moreover, to ensure consistency and reproducibility of the experiment, the random number seeds used in the code are set to 5328 or values derived from 5328.

Before running the code, we ensure *TensorFlow* and *Keras* are updated to the latest versions (*TensorFlow* version: 2.17.0; *Keras* version: 3.4.1) to guarantee the proper functioning of algorithms. Additionally, the warnings module ignores warnings of the UserWarning category, thereby reducing unnecessary clutter in the program output.

C. Execution Steps

The code operation is mainly divided into several parts: data reading, data segmentation, preprocessing, noise application, CNN algorithm settings, construction of the Expectation-Maximization (EM) algorithm, construction of three classifiers with different loss functions (Cross Entropy, Reweighted, and Generalized Cross Entropy Loss Function) based on the CNN model, and finally, evaluation of the algorithm using top-1 accuracy. First, in the code part, the information on the three data sets is printed. Through this, we found that CIFAR is not the traditional CIFAR-10 data set. The current data set only contains 4 categories, which is helpful for the subsequent estimation of its transposed matrix T. Then, the data of the three data sets are segmented (specifically 80% for training and 20% for validation) and normalized. Then the CNN model is built and trained. In addition, since the transposed matrix T of CIFAR.npz is unknown, the Expectation-Maximization (EM) algorithm is constructed in the fifth part to estimate the transposed matrix T of CIFAR.npz. And then, we constructed two loss function methods (Reweighted and Generalized Cross Entropy Loss Function) for building classifiers. Next, we built the evaluate_model method to evaluate the three classifiers, which includes 10 trainings for each classifier and then outputs a list of all test accuracy values. Finally, the output of the evaluate_model method is used to visualize the evaluation, including line charts, bar charts, and heat maps.