Analyzing the robustness of classifiers to label noise in image classification.

Jiarui Song
SID: 490043785
Unikey: json8547

Anh Dung Tu
SID: 500103247
Unikey: antu8283

Xiaojing Ren
SID: 530150770
Unikey: xren6548

Tutors: (Siyu Xu, Daniel Friedrich, Rafiul Nakib)

Abstract

This report investigated two label noise-robust classifiers on three datasets that have class-dependent label noise. The classification implemented are CNN with backward correction and generalized cross-entropy (GCE) loss, and residual network (ResNet) with importance reweighting. ResNet-18 neural network is used to estimate transition matrices. Among the three given datasets, FashionMNIST0.5, FashionMNIST0.6 are provided with transition matrices, while the transition matrix of the CIFAR dataset is not given. The obtained results suggests that the label noise-robust measures to add to these classifiers effectively mitigated the effect of label noise and improves the accuracies of classifiers for all three datasets. The transition matrix estimator that we utilized achieved high precision, with mean absolute error (MAE) less than 0.1. However, more complex structures and robust methods may be needed for the CIFAR dataset.

1 Introduction

A lot of real-world tasks are now automated with classifiers, which are a type of supervised machine learning models. An example is to recognize images and classify them into right labels. For example, digit recognition classifiers map images of digits to actual digits. Typically, classifiers in machine learning algorithms work well if their training dataset and test dataset have similar data distributions. However, the real-world data set is often complicated, filled with noise commonly categorized in two types: feature noise (attribute noise) and label noise (class noise) [1]. Feature noise refers to corruption in the features of the data, whereas label noise refers to wrong labels in the training data, or misclassification [1]. The latter is generally more detrimental, as multiple features typically exist in a data set, but only one label is assigned to each sample [2], [1]. Sometimes, with multiple features, noise in one or two less important features may not have a large impact on the classification results, but the same is not true for labels [1]. In this study, we investigate machine learning models that are robust to label noise. Specifically, we will look into the scenario where the training data contains label noise that is class-dependent, but the test data does not.

Label noise is generated in various scenarios in real life. One example is that companies such as Amazon hires employees to manually label datasets [2]. First, these employees may not have the relevant experience to label the data. Second, datasets are sometimes not informative enough and may be hard to recognize, such as blurry images, which can induce label noise [2]. Lastly, label noise can happen due to subjectivity of the labeling, where different experts have different opinions [2]. With the prevalence of label noise in real-life data, it is important to investigate machine learning models that generalize well to clean test data after training on label noise data.

Two types of label noise are random label noise and class-dependent label noise[1], [2]. Random label noise or noise completely at random (NCAR) [2] refers to a fixed probability of noise in labels of the dataset, not relying on classes. Class-dependent label noise or noisy at random (NAR) [2] means that the probability of noise in labels is different depending on the class of the sample. Label noise can also be correlated with the features, but in this project we do not address this type of label noise.

Label noise has various negative impact on different classifiers, which further motivates the investigation in label noise robust classifiers. Data with label noise require more complex model structures but still result in poorer classifier performance [2]. For example, with the presence of label noise, k nearest neighbors (kNN) classifier typically needs a higher optimal k. Intuitively, 1NN is the most susceptible to noise, and a higher k is more robust to label noise because kNN considers more samples and neighbors. Adaboost, on the other hand, tends to overfit the noise labels. In each step, the misclassified samples with wrong labels have increased weights, misleading the algorithm and decreasing the performance of the Adaboost algorithm on clean data.

Several methods improves the robustness of a classifier against label noise [2], [3]. Three main approaches include choosing a naturally robust classifier and loss function, preprocessing noisy labels in training data, or direct modeling of noisy labels during training process. Bagging and Random Forest are known to be more robust than Decision Trees or Adaboost [2]. 0-1 loss function and the mean absolute value of error (MAE) are also more robust to noisy labels than log-loss. K-nearest neighbors (KNN) or voting are used to remove noisy labels before training a classifier. Finally, transition matrix and forward (or backward) correction is employed in neural networks to model the inherent properties of label noise during the training process.

In this report, we intend to investigate the robustness of two label noise-robust classifiers: CNN with backward correction and generalized cross-entropy loss and ResNet with importance reweighting, across three datasets: Fashion-MNIST0.5, Fashion-MNIST0.6, and CIFAR. These datasets have class-dependent label noise. FashionMINIST contains grayscale images of tops, trousers, and dresses with given transition matrices. CIFAR contains colored images of cars, planes, and cats with unknown flip rates. The transition matrix of CIFAR will be estimated using ResNet-18. The transition matrix estimation results will be evaluated with mean absolute error(MAE) against the given transition matrices in FashionMNIST datasets. The classifier results will be evaluated with metrics such as F1-score, accuracy, recall and precision.

This report is organized as follows. Section 2 of this report will provide the problem statement and an overview of the methods commonly used for label noise-robust classification. In Section 3, we will introduce the transition matrix estimator we utilized and the two label noise-robust classifiers. In Section 4, we will present our experimental setting, evaluation metrics, hyperparameters chosen for models, experimental results and discussion. In Section 5, we present the conclusion and reflection for our project. In Section 6, we will discuss potential future work extending this project.

2 Related work

2.1 Problem Statement

Let D be a distribution of the pair of random variables $(X, Y) \in \mathcal{X} \times C$, where $\mathcal{X} \subseteq \mathbb{R}^d$ and $C = \{0, 1, 2\}$ refers to the set of possible classes. Let $(X_1, Y_1), \ldots, (X_n, Y_n)$ be an i.i.d. sample drawn from the distribution P and $(X_1, \hat{Y}_1), \ldots, (X_n, \hat{Y}_n)$ be the corresponding sample with label noise [4]. We assume that our label noise is class-dependent, meaning

$$P(\hat{Y} = i \mid Y = j) = \rho_{ji}, \tag{1}$$

where flip rates $\rho_{ij} \in [0, 1)$, $i, j \in C$ and $i \neq j$. The sum of flip rates is less than one $\sum \rho_{ij} < 1$ [4]. We let D_ρ be the distribution of the label noise variables $(X, \hat{Y})$.

The task is to find a classifier $f_n : \mathcal{X} \to C$ that trains from label noise variables $(X, \hat{Y})$ but correctly predicts $Y \in C$ in the test data.

In our project, for the Fashion-MNIST0.5 and Fashion-MNIST0.6 datasets, the transition probabilities ρ_{ij} are known; however, the flip rate ρ_{ij} for the CIFAR dataset is unknown.

2.2 Review of label noise methods

Several approaches to handle label noise exist in literature. This section discusses three of them as discussed in papers [2], [3].

The first group relies on label noise robust classifiers that are naturally robust to label noise. However, this approach does not specifically address label noise. In fact, reductions in label noise are achieved primarily through preventing overfitting. Therefore, often the effect of label noise is not completely neutralized [2]. The focus of these methods is to select robust classifiers, loss functions, and training procedures. It is well known that decision trees are highly sensitive to label noise, while random forests are more robust to it. Post-pruning trees, which is a common method to prevent overfitting in trees, mitigates the effect of label noise [2]. Bagging also performs better than boosting, because boosting increases the weights of misclassified noisy labels during the training process. In contrast, bagging benefits from the data variability introduced by label noise.

In addition to classifiers, the choice of loss functions further improves robustness against noisy labels. 0-1 loss and least-square loss are robust to uniform label noise, but not smoother functions such as log-loss, hinge-loss and exponential loss [3]. Several work have developed loss functions that are particularly robust to noisy labels. Essentially, label noise-robust loss function penalizes the model less severely for incorrectly predicting the data with wrong labels. Some examples are mean absolute value of error (MAE), generalized cross-entropy loss [5], and Bootstrap Loss [6]. In this project, we will also explore the Generalized cross-entropy (GCE) loss.

The second type of method preprocesses the label noise in the training data. Misclassified instances are generally relabeled or removed. These methods are fast and simple to use, but they can introduce bias and remove important data points [2], especially when the data is imbalanced or the label noise is not at random. One such method is to use voting among an ensemble of classifiers to identify mislabeled data[3]. K-nearest neighbors (KNN) can also help remove mislabeled samples. However, studies have shown that voting and KNN tend to remove too many data points. In addition, some more computational intensive methods use the impact of a data point on the classification result to determine if the particular data point is misclassified. Other approaches use techniques similar to outlier-detection and remove training data that exceed a threshold based on measures of the complexity of data points [3].

The third type of method directly models label noise and combines the model with classifiers [2], [3]. Hence, the inherent nature of the label noise is considered in the classification process. Two approaches are introduced for this type.

The first approach is to use a small clean dataset [3]. A classifier is trained on the noisy data first and is subsequently fine-tuned with the small clean dataset. Another method is to train an independent classifier on the clean data to remove noisy labels. A second classifier is subseqently trained on the noisy data to perform the main task. However, a small clean dataset may not always be available in real life [3]. Furthermore, incorporating additional fine-tuning or additional classifiers are computationally more expensive.

For the second approach, a ”noisy layer” is added to the end of deep learning network such as the convolutional neural network (CNN), to deal with noisy labels [3]. This approach, also known as loss correction, is explored in this report. In this method, the logits of the deep learning model are multiplied by a transition matrix (in the forward method), which represents the probabilities of misclassification. In contrast, the output probabilities of the model are multiplied by the inverse of the transition matrix in the backward method. However, the downside of this approach is that the transition matrix does not always exist in the real world and we often need to estimate it. The estimation of a transition matrix also proves to be challenging. Many studies had investigated transition matrix estimation extensively [7].

One approach to estimating transition matrices is anchor point estimate. Initially, anchor points for each class are estimated [8]. Anchor points are data points that are highly likely to be correctly labeled despite the presence of noise. At least one anchor point should exist for each class, so the likelihood of correct and wrong labels for each class can be inferred from their anchor points.

Different approaches to deal with label noise exist in literature. In this report, we will compare and investigate the robustness of two models: a CNN model with backward correction and generalized cross-entropy (GCE) loss and a ResNet model employing an importance reweighting technique. Evaluation will be done with F1-score, accuracy, recall and precision. The results will be benchmarked against a baseline CNN model, which does not incorporate any label noise-robust measures.

3 Methods

This section describes and compares the methods we used for transition matrix estimation and label noise-robust classifications. We used ResNet-18 to estimate the transition matrix for the CIFAR dataset. Two classifiers were utilized for the classification tasks: Importance reweighting combined with ResNet, CNN with backward correction and generalized cross-entropy (GCE) loss.

3.1 Transition matrix estimation with ResNet

A transition matrix describes the relationship between Y and $\hat{Y}$ [7]. In particular, $T_{ij} = P(\hat{Y} = i \mid Y = j)$. It is essentially a matrix of flip rates, representing the probability of misclassifying a datum as class i given the true class j, where $i \neq j$.

Transition matrix is typically used to add a ”noise layer” in deep learning methods. The output of the deep neural network is multiplied with transition matrices to mitigate the effect of noisy labels. Several methods exist to estimate a transition matrix, including T-revision [7] and anchor point estimate. In our project, we applied ResNet-18 to estimate the transition matrix of CIFAR.

To estimate the transition matrix using a ResNet-18 without relying on anchor points, we first train the ResNet on the noisy dataset. After training, the model's predictions on a validation set are analyzed to observe the distribution of predicted labels in comparison to the true noisy labels. By analyzing the patterns in the confusion between classes as reflected in the softmax outputs, we infer the probabilities of each true class being mislabeled as each predicted class. This analysis yields a transition matrix where each entry indicates the likelihood of misclassification between classes. The matrix is then normalized so that each row sums to one, reflecting the probability distribution of noise for each class. This transition matrix can be iteratively refined by integrating it into the training process of ResNet-18 to continually adjust for the label noise, and the final version is used to inform the loss function or to reweight samples in further training iterations to enhance the model's robustness against label noise.

We implemented ResNet-18 from the PyTorch library, which contains 18 layers grouped in 4 blocks. These blocks are composed of two convolutional layers, two batch normalization layers and a ReLU activation layer. The Figure 1 shows the ResNet-18 architecture in detail.

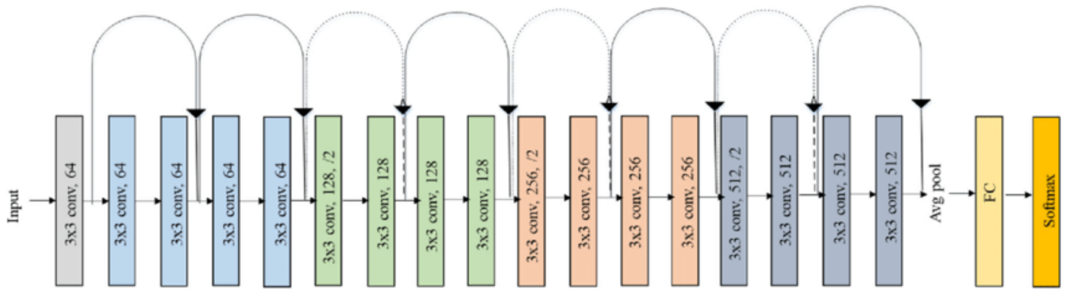


Figure 1: ResNet-18 Architecture [9].

3.2 Label noise-robust classification

We explore reliable techniques for categorizing multiple classes in images utilizing existing transition matrices or estimated transition matrices, and evaluate them based on top-1 accuracy, F1-score, recall, and precision. We delve into the Backward correction and the importance reweighting approaches. In this section, we provide an overview of each method's structure, the underlying theory supporting their robustness, along with the specifics we use to optimize these techniques.

3.2.1 CNN with backward correction and GCE loss

Overview The CNN model we employed contains two convolutional layers, each followed by a max pooling layer with a $2 * 2$ kernel and a stride of 2. Three fully connected (FC) layers are subsequently utilized, with a drop out layer inserted after the first FC layer. The drop out rate is tuned as a hyperparameter for each dataset.

The main idea behind backward correction [8] is to estimate the noise transition matrix and then utilize this matrix to adjust the output of a neural network. The noise transition matrix, often denoted by T, represents the probabilities of one class being mislabeled as another. The backward correction method will take the known transition matrix into the learning procedure with a noise adaption layer of the neural network. The noise adaption layer serves as a normal linear layer but has no bias and its weights will range between 0 and 1 because it represents the conditional probabilities T_{ij} . Please refer to Figure 2 for a visual explanation of backward correction.

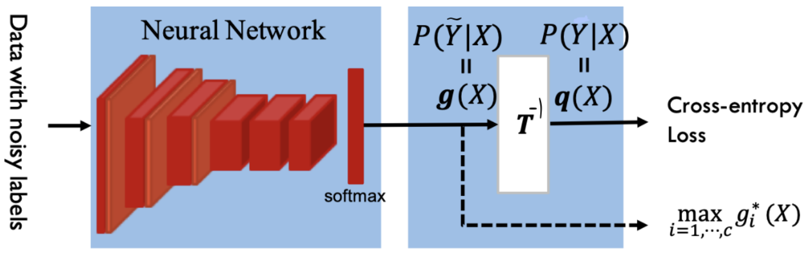


Figure 2: An explanation of the Backward correction.

The notations in Figure 2 represents the following elements:

- $\tilde{Y}$ - Observed (potentially noisy) label from the dataset.
- T^{-1} - Inverse noise transition matrix, where T_{ij} is the probability that a true label i is observed as j.
- Y - Unobserved true and clean label.
- $g(X)$ - The output of the softmax function and the predicted probabilities
- $q(X)$ - The corrected probabilities

Backward correction corrects the loss function to take into consideration the label noise flip rates in the transition matrix [8]. Assuming that the transition matrix is invertible and accurately representing the label noise in the data, backward correction proposes to adjust its loss accordingly to remove the effect of noisy labels. Because of this, the accuracy of transition matrices are paramount for backward correction method. In the actual implementation, the adjustment of the loss function is generally applied on the predicted probabilities. See Figure 2 for a visual explanation. The corrected probabilities are then used to calculate the loss against true labels.

Formulation Given:

- $\tilde{y}$ - Observed (potentially noisy) label from the dataset.

- T - Estimated noise transition matrix, where T_{ij} is the probability that a true label i is observed as j.
- $h(x)$ - The logits for the input x before applying the softmax function.
- $\hat{p} = \psi(h(x))$ - Predicted probabilities and the output of softmax function ψ.
- $\hat{p}'$ - Probabilities corrected by the inverse of the transition matrix T^{-1}.

The probabilities corrected by the transition matrix are defined as:

$$\hat{p}' = T^{-1}\hat{p} \tag{2}$$

Then, the loss is calculated with $\hat{p}'$:

$$l(h(x)) = l\left(T^{-1}\hat{p}\right) = l\left(T^{-1}\psi(h(x))\right) \tag{3}$$

Where:

- l is the loss function calculated using the corrected probabilities.
- T^{-1} is the inverse of the estimated transition matrix T.

The formula describes the backward correction: we first apply the softmax to the logits of the model to get the probabilities, then post-multiply these probability by the inverse of the transition matrix T to correct them, and finally compute the loss using the corrected probabilities. Since T is already known, we can minimize w.r.t the clean data distribution under the assumption that $\hat{P}(Y|X)$ approximates $P(Y|X)$. Luckily, all the given and estimated transition matrices are invertible for our datasets.

3.2.2 Importance reweighting with ResNet

Overview For robust classification model training, we redefine and make some adjustment to the structure of ResNet model from PyTorch, by implementing the ResNet architecture, including three initial convolutional layers, three residual layers, three fully connected layers (nine layers in total). By applying the ResNet architecture, the classifier can learn and adapt the complicated patterns in the datasets, especially CIFAR.

Formulation Importance reweighting [4] aims to adjust for the effects of potentially incorrect labels by introducing a weighting mechanism. This technique is based on the domain adaptation framework, which involves the transfer of knowledge from a source domain – in this instance, characterized by a noisy data distribution denoted by $\bar{D}$ – to a target domain, which is the 'clean' data distribution D. This method relies on the joint probability distribution of the variables (X, Y) across both the distributions D and $\bar{D}$.

The importance weight function, represented as $\beta(X, \hat{Y})$, is computed as the ratio of the probability of the pair (X, Y) in the true distribution D to the probability of the pair (X, $\hat{Y}$) in the noisy distribution $\bar{D}$, which is given by:

$$\beta(X, \hat{Y}) = \frac{P_D(X, Y)}{P_{\bar{D}}(X, \hat{Y})} = \frac{P_D(Y|X)}{P_{\bar{D}}(\hat{Y}|X)} \tag{4}$$

As the true distribution D is often not known, we use a known transition matrix T, assuming that $P_D(X) = P_{\bar{D}}(X)$, to empirically estimate the true distribution D .

3.2.3 Generalized cross-entropy loss

Overview We employed generalized cross-entropy loss [5] instead of cross-entropy loss for the CNN with backward correction model. The GCE loss function is known to be robust to label noise, and by adopting it, we slightly improved the result of our CNN model with backward correction. The GCE loss is effective against label noise, because it reduces the penalty for wrong classifications when there is a high uncertainty in the label of the data. Specifically, the loss function is non linearly associated with the probability and likelihood of the model's classification. When the probability of the model's prediction is high, or the confidence of the model for the true label in the training data is low, the loss is reduced. In other words, compared with standard cross-entropy loss, GCE loss gives more leniency towards the mistakes a classifier made if the model is confident in its own predictions.

Formulation The definition of GCE is following:

$$L_{\mathrm{GCE}}(q, y) = \frac{1 - q_y^\gamma}{\gamma}, \tag{5}$$

where:

- q is the predicted probabilities obtained from softmax.
- y represents the true labels.
- q_y is the probability the model assigned to the true class y.
- $\gamma \in (0, 1]$ is a hyperparameter that controls the extend of penalization for incorrect classifications. The smaller the γ, the more robust the loss is to label noise.

In standard cross-entropy loss, when q_y is low, the loss is penalized heavily, which is associated with the term $-\log q_y$ in the cross-entropy loss formula. However, for GCE loss, when the model is highly confident about its own predictions and assigned low probability to the true label in the data, the loss penalty is softened by the term $1 - q_y^\gamma$. The hyperparameter γ controls the extent to which the penalization is lenient towards the model's confidence in its classifications. This effectively mitigate the effect of label noise.

4 Experiments

4.1 Experimental setups

In the report, we used three datasets with class-dependent noisy labels, FashionMINIST0.5, FashionMINIST0.6 and CIFAR. Both FashionMINIST datasets contains 3 classes: tops, trousers, and dresses, while CIFAR contains airplanes, automobiles, and cats. See Figure 3 for example images of the datasets and Table 1 for a summary of the datasets.

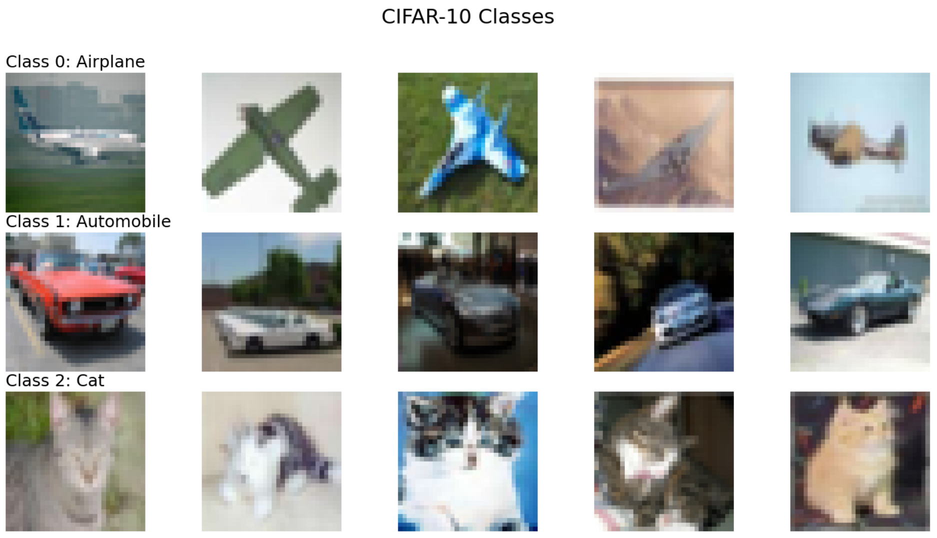
Dataset	Number of training	Number of test	Number of Classes	Image size
FashionMINIST05	18000	3000	3	28×28
FashionMINIST06	18000	3000	3	28×28
CIFAR	15000	3000	3	$32 \times 32 \times 3$

Table 1: Summary of datasets.

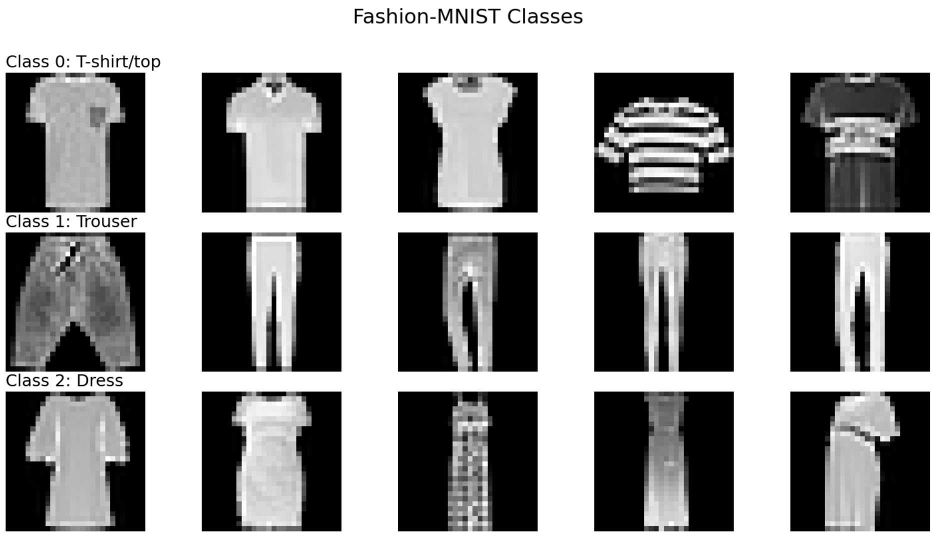
The noise transition matrix is provided for FashionMINIST datasets. The transition matrix for CIFAR is unknown, so we estimated the transition matrix with ResNet model. Both the given transition matrices and the estimated ones are listed in Table 2.

We employed two robust classifiers which are ResNet neural network with importance reweighting, a convolutional neural network (CNN) with generalised cross-entropy loss function in backward learning. The baseline model will be a CNN without learning transition matrix.

The classification results are evaluated with top-1 accuracy, recall, precision and F1-score. Transition matrix estimation is evaluated with mean absolute error (MAE). Details of these metrics are introduced in the Evaluation Metrics section below.



(a) CIFAR: cats, airplanes and automobiles



(b) FashionMINIST: Top, trousers and dresses

Figure 3: Example images of FashionMINIST and CIFAR

To rigorously evaluate the methods, training and validation are repeated 10 times by randomly sampling data from the training and validation/test set. Mean and standard deviation of all metrics are reported in Figure 4.

4.2 Evaluation Metrics

Different evaluation metrics are used to evaluate classification results and the estimated transition matrices. This section describes the evaluation methods we used for both tasks.

4.2.1 Transition matrix estimation evaluation metrics

To evaluate the performance of the transition matrix estimator, we use MAE, which computes the average of absolute differences between two matrices.

Mean Absolute Error

Given an estimated transition matrix $\mathbf{T}_{\text{estimated}} \in C \times C$ and the true transition matrix $\mathbf{T}_{\text{true}} \in C \times C$ where C is the set of classes, the mean absolute error, denoted as MAE, can be defined as:

$$\text{MAE} = \frac{1}{C^2} \sum_{i=1}^{C} \sum_{j=1}^{C} |\mathbf{T}_{\text{estimated}}(i, j) - \mathbf{T}_{\text{true}}(i, j)|$$

Where:

- $\mathbf{T}_{\text{estimated}}(i, j)$ is the element in the i^{th} row and j^{th} column of the estimated transition matrix.
- $\mathbf{T}_{\text{true}}(i, j)$ is the element in the i^{th} row and j^{th} column of the true transition matrix.
- C denotes the number of classes.

4.2.2 Evaluation metrics for classifiers

To evaluate the performance of the classifiers and their robustness, we use the Top-1 accuracy, Recall, Precision, and F1-score as the evaluation metrics. This section introduces these metrics.
Top-1 accuracy The formula for calculating Top-1 accuracy is:

$$\text{Top-1 Accuracy} = \frac{\text{number of correctly classified test samples}}{\text{total number of test sample}} \times 100$$

Number of correctly classified test samples can be interpreted as the number of test samples for which the class with the highest predicted probability matches the true class. The result is usually expressed as a percentage. A top-1 accuracy of 100% would mean that the model correctly predicted the class with the highest probability for all test samples.

Recall (or Sensitivity): Measures the proportion of actual positives that are correctly identified as such. It is especially important in situations where the cost of false negatives is high.

$$\text{Recall} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP) + False Negatives (FN)}}$$

Precision: Measures the proportion of identified positives that are actually correct. It's crucial in situations where the cost of false positives is high.

$$\text{Precision} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP) + False Positives (FP)}}$$

F1 Score: Harmonic mean of precision and recall. It seeks a balance between precision and recall.

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

4.3 Experimental results and discussion

4.3.1 Transition matrix estimation results

The transition matrix estimation is done with ResNet18. We evaluated the accuracy against given transition matrices from the FashionMNIST datasets. Figure 2 shows the original transition matrices of FashionMNIST datasts, the estimated transition matrices and their mean absolute error (MAE). It also contains the estimated transition matrix for CIFAR.

The transition matrix estimation achieved high accuracy on the FashionMNIST datasets, based on MAE. However, the classification results on CIFAR is not as good as FashionMNIST. It is possible that the transition matrix for CIFAR is not estimated well, resulting in poor classification results for CIFAR dataset. A relatively simple CNN model may also be the reason for the poor performance on CIFAR dataset.

Dataset	T	$\hat{T}$	MAE
Cifar10	-	$\begin{bmatrix}0.7945 & 0.0527 & 0.1528\\ 0.1472 & 0.6585 & 0.1943\\ 0.0799 & 0.0359 & 0.8842\end{bmatrix}$	-
FashionMNIST0.5	$\begin{bmatrix}0.5 & 0.2 & 0.3\\ 0.3 & 0.5 & 0.2\\ 0.2 & 0.3 & 0.5\end{bmatrix}$	$\begin{bmatrix}0.5105 & 0.2344 & 0.2551\\ 0.2798 & 0.5249 & 0.1953\\ 0.2383 & 0.3080 & 0.4537\end{bmatrix}$	0.0258
FashionMNIST0.6	$\begin{bmatrix}0.4 & 0.3 & 0.3\\ 0.3 & 0.4 & 0.3\\ 0.3 & 0.3 & 0.4\end{bmatrix}$	$\begin{bmatrix}0.5269 & 0.3117 & 0.1614\\ 0.3771 & 0.4553 & 0.1675\\ 0.4076 & 0.3385 & 0.2540\end{bmatrix}$	0.09269

Table 2: Comparison between true transition matrices and the estimated transitions matrices with ResNet.

4.3.2 Classification results and discussion

This section presents the classification results of our experiments. The two methods were trained repeatedly for 10 times in randomly sampled training and validation, implemented with the `KFold` class from the `scikit-learn` library. The result achieved indicates that both implemented classifications with two label noise correction methods improve the performance of the models without correction loss function, especially on two FashionMNIST datasets. Details of the result are discussed below.

CNN with backward correction

Hyperparameters Hyperparameters are tuned for each dataset with 1000 training samples. A total number of 50 randomly sampled hyperparameter groups are tested on the validation data, which are split from the training data. The 1000 samples are sampled in such a way that it has the same class distribution as the original training data. The hyperparameter values are sampled from the following lists:

- learning rates: $\{0.0005, 0.0001, 0.00001, 0.00005, 0.000001\}$
- batch sizes: $\{16, 32, 64, 128\}$
- GCE ratios: $\{0.1, 0.3, 0.5, 0.7, 0.9\}$
- dropout rates: $\{0.1, 0.2, 0.3, 0.4, 0.5\}$

The final hyperparameters used for each dataset are listed in Table 3.

Dataset	learning rate	drop out rate	GCE ratio	batch size
FashionMINIST05	1×10^{-5}	0.3	0.7	64
FashionMINIST06	1×10^{-5}	0.5	0.95	32
CIFAR	1×10^{-5}	0.5	0.6	64

Table 3: Hyperparameters for backward correction with CNN

Result The results of the CNN with backward correction classification can be found in Figure 4 and Table 4, 5. In summary, the model improved the results of the CNN baseline model in all three datasets. Although the accuracy for CIFAR is only around 60%, the classifier achieved accuracy greater than 0.92 for FashionMNIST0.5 and greater than 0.88 for FashionMNIST0.6.

Mysteriously, all recalls and accuracies are the same for all 10 experiments for each dataset. After investigation, we decided that it might be the nature of the datasets that resulted in this scenario.

Apart from that, the FashionMNIST dataset achieved high accuracy, greater than 0.85 for all experiments, but our classifiers performed less well on CIFAR, achieving accuracy less than 0.62 on all experiments. See Figure 4 for experimental results. Furthermore, CNN with backward correction achieved lower standard deviation when classifying FashionMNIST images. The standard deviation is higher for the CIFAR dataset, and thus the performance is less stable. It seems that our model did a better job for FashionMNIST datasets. See Figure 4 and Table 5 for the standard deviation results.

Dataset	Accuracy	Precision	Recall	F1-Score
FashionMNIST0.6	0.9246	0.9275	0.9246	0.9235
FashionMNIST0.5	0.8855	0.8905	0.8855	0.8805
CIFAR	0.5896	0.6259	0.5896	0.5592

Table 4: Mean Metrics for Each Dataset for CNN Backward Correction

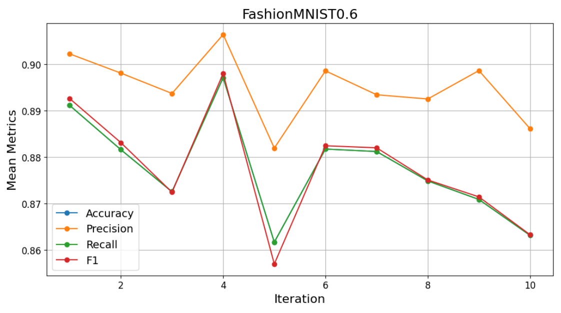
Dataset	Accuracy STD	Precision STD	Recall STD	F1-Score STD
FashionMNIST0.6	0.005	0.005	0.005	0.004
FashionMNIST0.5	0.004	0.002	0.004	0.003
CIFAR	0.01	0.025	0.01	0.015

Table 5: Standard Deviation of Metrics for Each Dataset for CNN Backward Correction

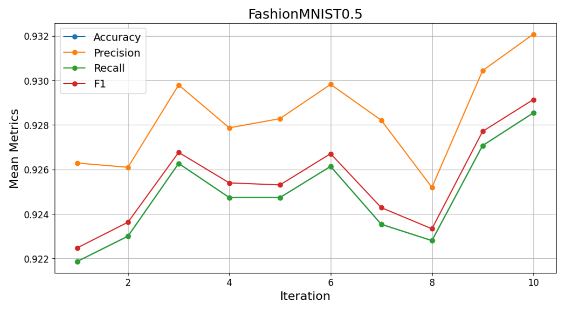
Table 6 also contains a comparison between the accuracies of the baseline model and the average accuracies achieved by CNN with backward correction model. The baseline model has the exact structure as our CNN model, but without the robust label noise measure: backward correction and GCE loss. For both the FashionMNIST0.6 and CIFAR dataset, the improvement is quite substantial. The model also improved the accuracy of FashionMNIST0.5 from 0.89 to 0.92 despite the strong performance CNN baseline model already achieved. In fact, for all three datasets and in all ten experiments, the accuracies achieved by our label noise-robust classifiers are higher than those of our baseline model, as shown in Figure 4.

Dataset	CNN with backward correction and GCE	CNN baseline
FashionMNIST05	**0.925**	0.89
FashionMNIST06	**0.88**	0.69
CIFAR	**0.59**	0.48

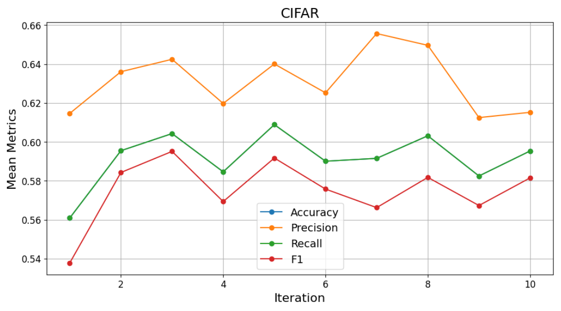
Table 6: Average accuracy achieved by CNN with backward correction and GCE and CNN baseline model.



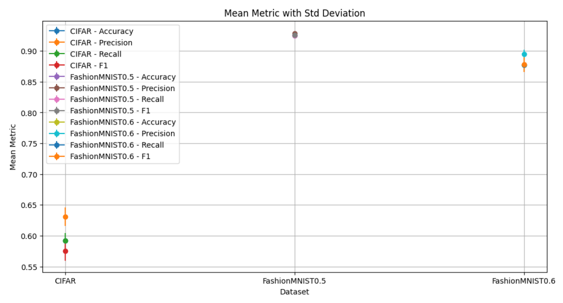
(a) FashionMNIST0.6 mean accuracy, mean F1-score, mean recall and mean precision for 10 experiments



(b) FashionMNIST0.5 accuracy, F1-score, recall and precision for 10 experiments



(c) CIFAR accuracy, F1-score, recall and precision for 10 experiments



(d) Standard deviation for FashionMNIST0.5, FashionMNIST0.6 and CIFAR over ten experiments

Figure 4: The mean and standard deviation of f1-score, accuracy, recall, and precision for CNN with backward correction

Discussion Our classifier does not achieve high accuracy on CIFAR. There could be various reasons. It is possible that the colored images in CIFAR are more complex to model, or that CIFAR is a smaller dataset than FashionMNIST (15,000 images versus FashionMNIST0.6's 18,000 images), so our neural network cannot fully model the patterns with the limited data. It is also possible that the transition matrix for CIFAR is not well estimated, resulting in poor performance.

Our label noise-robust measures successfully modeled the datasets despite the presence of label noise. It is obvious that our model did a good job in improving the performance of the classification comparing to the CNN baseline model. Although the improvement of FashionMNIST0.5 is small with accuracy enhanced by 0.03, but CNN baseline model already achieved 0.89 accuracy without any robust measure against label noise. The accuracy for CIFAR is also enhanced from 0.48 to 0.59, despite the poor performance.

ResNet with the Importance Reweighting Technique

Result We performed a single fold of 10-fold cross-validation, alongside a baseline for comparison. The enhancement of accuracies against the baseline is substantial. Table 7 shows for the FashionMNIST0.5 dataset, the model outperformed the baseline and CNN with the backward correction model, reaching the highest accuracy of 0.934. The improvement in the FashionMNIST0.6 dataset was also substantial, with the model achieving 0.873, whereas the baseline is only 0.69. However, compared to the baseline model, the classifier experienced a decrease in performance in the CIFAR dataset.

Dataset	Highest Accuracy (10-fold CV)	Baseline
FashionMNIST0.5	**0.934**	0.89
FashionMNIST0.6	**0.873**	0.69
CIFAR	**0.486**	0.48

Table 7: ResNet with Importance Reweighting Loss Function, showing the highest accuracy in a single fold of 10-fold cross-validation.

Table 8 and Table 9 showcase each dataset's mean metric and standard deviation. The mean accuracy for FashionMNIST0.5 is high at 0.9110, with similarly high precision and F1-score, suggesting a well-fitted model. For FashionMNIST0.6, all metrics show competent performance, although there is a slight drop in accuracy and F1 score compared to FashionMNIST0.5. The standard deviations for both FashionMNIST datasets are quite low, indicating that the model's performance is consistent across different cross-validation folds.

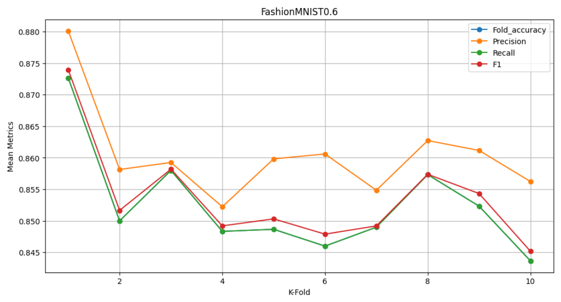
However, for CIFAR the mean metrics show that the model struggles, with a mean accuracy of 0.4301 and a particularly low recall of 0.4301. The standard deviations, especially for precision and F1-score, are high relative to the means, which indicates considerable performance variability across CV folds.

Dataset	Accuracy	Precision	Recall	F1-Score
FashionMNIST0.6	0.8526	0.8605	0.8526	0.8537
FashionMNIST0.5	0.9110	0.9167	0.9110	0.9111
CIFAR	0.4301	0.6383	0.4301	0.3474

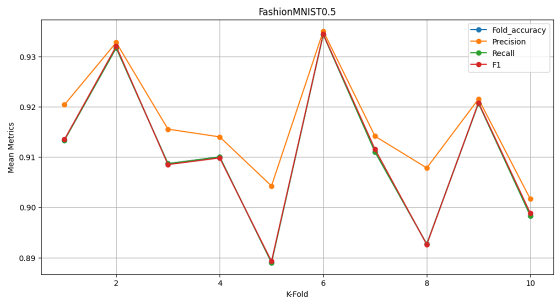
Table 8: Mean Metrics for Each Dataset

Dataset	Accuracy STD	Precision STD	Recall STD	F1-Score STD
FashionMNIST0.6	0.0079	0.0072	0.0079	0.0078
FashionMNIST0.5	0.0143	0.0105	0.0143	0.0144
CIFAR	0.0366	0.0833	0.0366	0.056

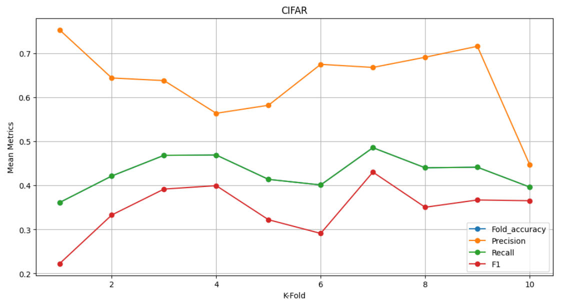
Table 9: Standard Deviation of Metrics for Each Dataset



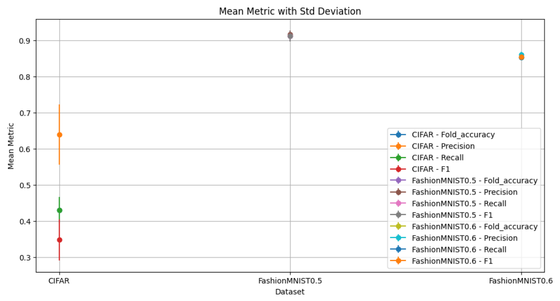
(a) FashionMNIST0.6 mean accuracy, mean F1-score, mean recall and mean precision for 10 experiments



(b) FashionMNIST0.5 accuracy, F1-score, recall and precision for 10 experiments



(c) CIFAR accuracy, F1-score, recall and precision for 10 experiments



(d) Standard deviation for FashionMNIST0.5, FashionMNIST0.6 and CIFAR over ten experiments

Figure 5: Results (f1-score, accuracy, recall, and precision) for ResNet with Importance Reweighting Loss

Discussion From integrating the results from all three tables into Figure 5, we learned the Importance Reweighting Loss Function has a better performance on the less complex FashionMNIST datasets. Furthermore, the increased complexity and variability of the CIFAR dataset suggests more sophisticated data augmentation, regularization techniques, and potentially a deeper or differently structured network to achieve higher performance.

Comparison and discussion

Furthermore, comparing the results of the two classifiers, both models achieved a high average accuracy, exceeding 0.91, on FashionMNIST0.5 and performed poorly on the CIFAR dataset, with mean accuracy below 0.6. Both models achieved low standard deviation in accuracy on the FashionMNIST0.5 dataset across 10 experiments, suggesting stability and consistency of the model's performance on FashionMNIST. Overall, the results suggest difficulties both classifiers have predicting the CIFAR dataset, because the classifiers have the lowest accuracy and highest standard deviation for predicting CIFAR.

4.4 Hardware

The development of the code was carried out using Python 3.x and Jupyter Notebook, leveraging the computational resources offered by Google Colab. We run the experiments using GPU instead of CPU, resulting in significantly shorter run time, thus it is highly recommended to use GPU when running the notebook.

5 Conclusion

In this project, we investigated two classifiers designed to enhance the robustness of the model against label noise. The classifiers are CNN with backward correction and GCE loss and ResNet with importance reweighting techniques. The three datasets we utilized are FashionMNIST0.5, FashionMNIST0.6, and CIFAR. The CNN model used the transition matrices given from the FashionMNIST datasets and the estimated transition matrix of CIFAR in the training process. The estimation of the transition matrix of the CIFAR dataset is done with ResNet-18.

It was evident through observation that the transition matrix estimation using ResNet-18 algorithm delivers highly precise outcomes. This was ascertained by evaluating its performance against the provided transition matrices for the FashionMNIST0.5 and FashionMNIST0.6 datasets. However, our estimator might not remain its high performance when it comes to CIFAR, since our classifier did not perform well with these colored images.

The classification results of both classifiers showed substantial improvements on all three datasets in terms of accuracy against the CNN baseline model. Both classifiers achieved relatively good accuracies on FashionMNIST datasets, improving the results of the CNN baseline model, but performed less well on CIFAR. Please see Table 6, 7 for the summary of results.

Both methods have poor performance on the CIFAR dataset which contains color images, achieving accuracy less than 60%. We speculate two possible reasons: first, it is likely that the model complexity is not capable of capturing patterns as well as learning the label noises in more complex data such as CIFAR. Second, the estimated transition matrix may not accurately represent the label noise distributions in the data. Further experiments and research could address this issue.

6 Future Work

For future works, we can extend the research and implementation of this project following these directions:

- Implement more robust method such as T-revision, Forward learning or Bootstrap loss function as well as ensemble classification methods that can improve the classification results with the presence of label noise.
- Investigate other neural network structures in transition matrix estimation to improve transition matrix estimation results.
- Extend the work to other datasets such as CIFAR10 and CIFAR100, which have more classes. We may validate our approach on more complex datasets

References

[1] X. Zhu and X. Wu, "Class Noise vs. Attribute Noise: A Quantitative Study."

[2] B. Frenay and M. Verleysen, "Classification in the Presence of Label Noise: A Survey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 5, pp. 845–869, May 2014. [Online]. Available: http://ieeexplore.ieee.org/document/6685834/

[3] D. Karimi, H. Dou, S. K. Warfield, and A. Gholipour, "Deep learning with noisy labels: Exploring techniques and remedies in medical image analysis," *Medical Image Analysis*, vol. 65, p. 101759, Oct. 2020. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1361841520301237

[4] A. Díaz and D. Steele, "Analysis of classifiers robust to noisy labels," Jun. 2021, arXiv:2106.00274 [cs]. [Online]. Available: http://arxiv.org/abs/2106.00274

[5] Z. Zhang and M. Sabuncu, "Generalized Cross Entropy Loss for Training Deep Neural Networks with Noisy Labels," in *Advances in Neural Information Processing Systems*, vol. 31. Curran Associates, Inc., 2018. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2018/hash/f2925f97bc13ad2852a7a551802feea0-Abstract.html

[6] S. Reed, H. Lee, D. Anguelov, C. Szegedy, D. Erhan, and A. Rabinovich, "Training Deep Neural Networks on Noisy Labels with Bootstrapping," Apr. 2015, arXiv:1412.6596 [cs]. [Online]. Available: http://arxiv.org/abs/1412.6596

[7] X. Xia, T. Liu, N. Wang, B. Han, C. Gong, G. Niu, and M. Sugiyama, "Are Anchor Points Really Indispensable in Label-Noise Learning?" Dec. 2019, arXiv:1906.00189 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1906.00189

[8] G. Patrini, A. Rozza, A. Menon, R. Nock, and L. Qu, "Making Deep Neural Networks Robust to Label Noise: a Loss Correction Approach," Mar. 2017, arXiv:1609.03683 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1609.03683

[9] F. Ramzan, M. U. G. Khan, M. A. Rehmat, S. Iqbal, T. Saba, A. Rehman, and Z. Mehmood, "A deep learning approach for automated diagnosis and multi-class classification of alzheimer's disease stages using resting-state fmri and residual neural networks," *Journal of Medical Systems*, vol. 44, no. 2, p. 37, 2019.

A Appendix

A.1 Run code

Please run our code on GPU. The hardware we used is listed in Experiment section. Please place the datasets in the .npz format in the working directory and use the package versions listed below.

Package	Version
numpy	1.19.2
pandas	1.1.3
matplotlib	3.3.2
scikit-learn	0.23.2
tensorflow	2.3.0
keras	2.4.3

Table 10: Versions of the packages used in experiments.

A.2 Contribution

Each member of our team has made an equivalent and significant contribution to this project.