Label Noise Robust Classifiers

Fabiola Bella Wibowo
fwib8777@uni.sydney.edu.au

Nicole Thirza Wijaya
nwij0698@uni.sydney.edu.au

Abstract

Label noise has been demonstrated to hinder the training process of machine learning models, casting doubt on the accuracy of supervised algorithms when faced with such noise. In this report, we try to design three robust classifier algorithms resilient to label noise. Two of these classifiers, the noise adaptation layer and forward loss correction, are constructed by training with a transition matrix that estimates the flip rate of noisy labels for each label in the training set. In contrast, the third robust algorithm, rank pruning, avoids the use of the transition matrix. Instead, it trains models on 'confident' samples by pruning the original training set of inconsistent samples. The study assesses the performance of these three classifiers on three datasets. FashionMNIST0.5 and FashionMNIST0.6 are provided with pre-estimated transition matrices. The third dataset, CIFAR, features an unknown transition matrix, which we estimate using the average label transition probability acquired from a prior transition matrix estimated from a set of anchor points. The effectiveness of these algorithms in the presence of label noise is measured using four evaluation metrics: top-1 accuracy, precision, recall, and F1 score. Rank pruning achieves the best results and consistency for both FashionMNIST0.6 and CIFAR datasets, while the noise adaptation layer works best for FashionMNIST0.5 where noise rates are lower.

1 Introduction

Learning with label noise is a widely studied topic [3][5]. Noisy labels have been proven to degrade the performance of many machine learning algorithms, even more so than feature noise [5]. The need to collect data at a large scale to increase the performance of machine learning algorithms causes many large-scale datasets to be collected through crowdsourcing platforms, online queries, and image engines where label noise is inevitable due to amateur human annotators [4]. It is time-consuming and expensive to clean datasets of their noisy labels. In contrast, if we build a machine learning algorithm in such a way that it can learn with noisy labels without the noisy labels hampering its ability to predict the true clean labels, we can theoretically disregard the presence of noisy labels in any dataset up to a certain rate. We then can see how crucial it is to build machine learning algorithms that are resistant to label noise.

Despite the remarkable performance of deep learning methods, such as Convolutional Neural Networks (CNNs), in various applications, they remain susceptible to label noise. The expressive power and numerous parameters of CNNs make them prone to overfitting noisy labels, especially when dealing with large datasets. To address this vulnerability, adjustments are necessary to enhance model robustness.

In this paper, we construct several CNN architectures and seek to enhance their robustness using three different methods: the noise adaptation layer, loss correction approach, and rank pruning. Both the noise adaptation layer [10] and the forward loss correction [16] approach consider the dataset's transition matrices in their calculations to better adjust the machine learning models to label noises. In noise adaptation layer, we add an additional output layer on top of our original architecture to modify the output according to the given transition matrix. In forward loss correction, we correct the output layer's loss value by adjusting the loss function according to the transition matrix.

However, a limitation of these algorithms is their dependency on knowing the transition matrix. In response, we adopt a different approach in building our machine learning model using rank pruning [9], where the model is trained on samples with predicted probability of being positive near 1 when the label is positive or 0 when the label is negative. This algorithm directly contrast modern view of big data where it emphasize the value of more examples for better training.

In this report, we compare all the models with each other, using original CNN with cross-entropy loss function as the baseline. In order to determine the most robust classifier for this classification task, the performance of each model will be evaluated using four different methods: Top-1 Accuracy, Recall, Precision, and F1-Score.

2 Related Work

Label noise can be grouped into two types: feature-independent label noise, and feature-dependent label noise [5]. Feature-dependent label noise is a more realistic assumption of the label noise as it considers the features of the training data in the probability of a label being mislabeled. However, this type of label noise is much more difficult to estimate a reliable label of [3], and requires a complex modelling to solve [5]. Hence, for many noisy label robustness methods, an asymmetrical noise rate in feature-independent label noise is the preferred assumption to make.

A symmetrical feature-independent label noise (also known as random classification noise) assumes that each probability of true label Y being flipped into any wrong label $\tilde{Y}$ holds the same value. In contrast, an asymmetrical feature-independent label noise (also known as class-dependent label noise) assumes that each label that true label Y can be wrongly flipped to holds a different probability value. These probabilities are then modeled as a transition matrix [3], such as:

$$T = \begin{bmatrix} P(\tilde{Y}=0|Y=0,X) & \dots & P(\tilde{Y}=0|Y=C,X) \\ \dots & \dots & \dots \\ P(\tilde{Y}=C|Y=0,X) & \dots & P(\tilde{Y}=C|Y=C,X) \end{bmatrix} \tag{1}$$

We can see that true labels occur diagonally. In cases where there are no noisy labels, the transition matrix becomes an identity matrix.

Due to the complexity of feature-dependent label noise, we deem the asymmetrical feature-independent label noise to be an appropriate estimation method for the noise rate of a given training sample. The transition matrix produced to model the asymmetrical label noise then can be utilised by several robust classifier methods.

Importance reweighting is a method widely used in transfer learning to maximize a machine learning's performance in a certain target domain when trained with a different source domain [15]. Applying this to the classification problem in the presence of label noise, the method takes in a transition matrix in order to calculate weights to parameterize estimated true labels and estimated noisy labels. The weight is then formulated by the following, with $\rho_0^k = P(\tilde{Y}^k \neq 1|Y^k = 1)$ and $\rho_1^k = P(\tilde{Y}^k = 1|Y^k \neq 1)$ [15]:

$$\beta(X,\tilde{Y}) = \frac{(P^k|X) - Y^k \cdot \rho_1^k - (1 - Y^k) \cdot \rho_0^k}{(1 - \rho_1^k - \rho_0^k)P(\tilde{Y}^K|X)} \tag{2}$$

Although this method is designed to be generic and applicable to any machine learning algorithm [15], in practice it can be rather difficult to implement as it tends to need a separate weighting function to perform the calculation of $\beta(X,\tilde{Y})$ [5]. In regards to its projected complexity, we examine another label noise robust method that utilises the transition matrix.

The **backward loss correction** initially trains the model with the noisy label. After initial loss is acquired, subsequent steps will correct the loss by multiplication of the inverse of estimated transition matrix $\tilde{T}^{-1}$ [16]. The backward loss can be formulated as the following [5]:

$$\overleftarrow{\ell}(f(x;\theta),\tilde{y}) = \tilde{T}^{-1} < \ell(f(x;\theta),1),\ldots,\ell(f(x;\theta),C) >^{T} \tag{3}$$

The other method proposed in block [16] is the **forward loss correction** method. As described in block [16], the forward loss correction corrects the model's loss value by multiplying the neural network's prediction by T. This premise is similar to another method proposed by block [17], where they add a constrained linear layer on top of the softmax output layer to change the neural network's prediction to match the given noise rate. The objective function of the **linear noise adaptation layer** then is to maximize the following log likelihood of N samples [17]:

$$L(\theta) = \frac{1}{N}\sum_{i}^{N}\log(\sum_{i} q_{\tilde{y}_n i}\, p(y^* = i|x_n,\theta)) \tag{4}$$

Where θ denotes parameters of the neural network and Q denotes the given transition matrix. Considering that the linear adaptation layer is constrained to produce a probabilistic output, we pondered how the noise adaptation layer would perform when set as a softmax layer instead. Due to the flexibility, simplicity, and similarity of premises of both forward loss correction and noise adaptation layer, both methods are chosen for our label noise robustness method.

Finally, we looked into label noise robustness methods that did not utilise the transition matrix. In **label smoothing**, one-hot encoded labels are mixed with uniform label vectors to help the performance of machine learning algorithms with the generalization problem [19]. Another example is **bootstrapping**, where the prediction of an example is done by comparing the consistency of its features against other examples with similar features [20]. We have decided to include one of such method in order to roughly compare the performance of label noise robust methods when they don't adhere to the typical model of an asymmetrical feature-independent label noise.

3 Methods

3.1 Flip Rate Estimation

The transition matrix, also called a flip matrix, represents class posterior probabilities for clean data [5]. It is of C x C size, with C representing the number of classes we have in our target label. Utilising the product and sum rule, we can construct the following example formulation:

$$\begin{bmatrix} P(\tilde{Y}=0|X) \\ \ldots \\ P(\tilde{Y}=C|X) \end{bmatrix} = \begin{bmatrix} P(\tilde{Y}=0|Y=0,X) & \ldots & P(\tilde{Y}=0|Y=C,X) \\ \ldots & \ldots & \ldots \\ P(\tilde{Y}=C|Y=0,X) & \ldots & P(\tilde{Y}=C|Y=C,X) \end{bmatrix} \begin{bmatrix} P(Y=0|X) \\ \ldots \\ P(Y=C|X) \end{bmatrix} \tag{5}$$

Where $\tilde{Y}$ represents the observable noisy label and Y represents the unknown clean label, establishing that by learning the transition matrix we would be able to infer the clean class posterior. This can be done simply if we have reliable data, i.e anchor points that we can confidently estimate to belong to a clean label. Since the clean labels are unknown, we can instead estimate these anchor points. When upper-bounded by a constant (typically $[0, 0.5]$), we can estimate the anchor points by the following equation [7]:

$$x^i = argmax_{x\in\mathcal{X}} P(\tilde{Y}=i|X=x) \tag{6}$$

This is our first method for the flip rate estimation. From these estimated anchor points we can construct the initial transition matrix. The second method entails taking the initial transition matrix and averaging out the values for each label's transition probability into another label. The formulation for this method can be written as follows [5]:

$$\tilde{T} = \frac{1}{C} \sum_{i=1}^{i} \sum_{i \neq 1}^{j} \tilde{T}_{ij} \tag{7}$$

3.2 Convolutional Neural Network

Inspired by visual perceptrons [11], CNN is a Neural Network (NN) architecture that utilizes a convolution layer in its architecture to produce feature maps. The number of feature maps produced depends on the number of filters used to capture the features of the training samples; one filter would produce one feature map, two filters would produce two feature maps, and so forth. Thus, depending on the parameters used to build the CNN model, the features captured can be large in size and thus prone to overfitting. To overcome this, a pooling layer is often added after each convolution layer to "summarize" the output of the convolution layer.

For this experiment, we will be constructing three different CNN architectures to fit onto the three datasets. This is done so that we can obtain three models that work very well on each dataset and have a definite judgement on whether our robust methods can improve the model's robustness against the noisy label or not. We perform parameter grid search to acquire best performing models for each dataset. The architectures the grid search yielded are as follows:

Table 1: CNN architecture for each dataset, yielded from parameter grid search tuning

CNN Model	Convolution Layers	No. of Filters	Activation Function	Batch Size
FashionMNIST0.5	2	64	ReLu	64
FashionMNIST0.5	3	32	Tanh	128
CIFAR	2	64	Tanh	128

For all three of our models, we opted for the sparse categorical cross-entropy loss function, given the multiclass classification nature of the task. However, in the case of the loss correction method, a custom loss function was employed. Due to time constraints, we selected an adaptive learning optimizer, specifically choosing the Adam optimizer for its renowned performance and its ability to achieve local convergence under specific conditions [12]. Additionally, we set the number of epochs to five hundred for each test. To mitigate overfitting, a fifty percent neuron dropout rate was implemented, along with an early stopping method with a patience of 5 epochs.

3.2.1 Noise Adaptation Layer

The noise adaptation layer method adds an additional output layer on top of an existing CNN architecture, directly modifying the architecture's baseline output in accordance to the given transition matrix. The noise adaptation layer we implement is designed by block [10], which is a softmax noisy adaptation layer that connects the baseline output to the most probable noisy class candidate in accordance to the transition matrix.

This method is based on the Expectation-Maximization (EM) which objective is to maximize likelihood estimation in the presence of latent variables. The E-step is to estimate the latent variables and the M-step is to optimize the model. Iterating between the EM algorithm to estimate the noise rate and actually training the neural network is computationally expensive however, so the maximum likelihood scoring is implemented directly within the framework of the neural network instead. Block [10] proves that the additional softmax layer has the exact same formulation as an EM algorithm, making optimization more efficient as we are optimising a singular model instead of iterating between the two algorithms.

Block [10] then denotes the softmax layer that predicts the true label y by the following equation, with non-linear function performed on an input x as $h = h(x)$ and w as weights of the neural network:

$$p(y=i|x;w) = \frac{\exp(u_i^T h + b_i)}{\sum_{l=1}^{k} \exp(u_i^T h + b_i)} \tag{8}$$

3.2.2 Forward Loss Correction

We are familiar with the loss function. The loss function measures the error rate of a given machine learning algorithm which in turn is used to determine how the machine learning algorithm should act upon its given training sample. For example, the Mean Square Error (MSE) squares the error rate while Mean Absolute Error (MAE) takes the absolute value of the error rate. Using the MSE loss function would make a machine learning algorithm more susceptible to outliers as the error rate would have quadratic values, as opposed to the MAE loss function where the error rate is strictly only the unsigned distance between the true label and estimated label.

As it name suggests then, the forward loss correction method multiplies the baseline output of the softmax layer by an estimated transition matrix $\tilde{T}$ to correct the output layer's loss value. The formulation of the adjusted loss function, given that the initial loss function is cross-entropy, can be written as follows [16]:

$$\ell(e^i, \hat{p}(y|x)) = -\log \sum_{j=1}^{c} T_{ij} \hat{p}(y = e^j|x) \tag{9}$$

Where e^i represents a canonical vector in $\mathbb{R}$, $\hat{p}(y|x))$ denoting the class-conditional probability, and T as the transition matrix.

3.2.3 Rank Pruning

Rank pruning is an algorithm optimization targeted to learn noisy labels in the dataset. It's composed by two parts, estimation of the asymmetric noise rates and removal of mislabeled examples prior to training. In other words, we 'clean' the dataset out of mislabeled labels in order to better train our model.

The theoretical foundation of rank pruning is derived from $\tilde{P}\tilde{N}$ learning. Given a binary classification task where there exists positive labelled set and unlabelled set, the $\tilde{P}\tilde{N}$ learning process enforces a positive and negative constraint on the learning of the unlabelled set [18].

In rank pruning, given a binary classification task, estimation of confident and noisy data are constrained by an upper bound and lower bound, forming four sets of data. The size of the four sets of data are used to estimate the noise rates ρ_1, ρ_0, π_1, and π_0. The noise rate $\rho_1 = P(s=0|y=1)$ denotes the probability of positive examples mislabeled as negative, while $\rho_0 = P(s=1|y=0)$ denotes the probability of negative examples mislabeled as positive. Meanwhile, π_1 and π_0 are inverse noise rates derived from ρ_1 and ρ_0:

$$\hat{\pi_1} = \frac{\hat{\rho_0}}{p_{s1}} \frac{1 - p_{s1} - \hat{\rho_0}}{1 - \hat{\rho_1} - \hat{\rho_0}},\ \hat{\pi_0} = \frac{\hat{\rho_1}}{1 - p_{s1}} \frac{p_{s1} - \hat{\rho_0}}{1 - \hat{\rho_1} - \hat{\rho_0}} \tag{10}$$

Once the noise rates π_1 and π_0 are established, estimated noisy examples are removed from the training set, and the classifier is re-trained on the pruned dataset with reweighted samples. Positive samples has the weight of $\frac{1}{1-\hat{\rho_1}}$, while negative samples has the weight of $\frac{1}{1-\hat{\rho_0}}$.

The loss function of the rank pruning method can be formulated as below with s as observed noisy label, $\hat{y}$ as predicted true label, $\tilde{P}$ and $\tilde{N}$ as positive and negative noisy label sets, and finally ρ_0 as the fraction of mislabeled true N examples and ρ_1 as the fraction of mislabeled true P examples [9]:

$$\tilde{\ell}(\hat{y_i}, s_i) = \frac{1}{1 - \hat{\rho_1}} \cdot \mathbb{1}[[x_i \in \tilde{P}_{conf}]] + \frac{1}{1 - \hat{\rho_0}} \cdot \mathbb{1}[[x_i \in \tilde{N}_{conf}]] \tag{11}$$

3.3 Evaluation Metrics

We will be using four evaluation metrics to measure the robustness and performance of our robust classifier methods. The metrics we use are chosen to allow us insight into how each model performs in classifying each label. The **top-1 accuracy score** measures how many times the model correctly assigned the highest probability value to a correct label given a training input:

$$Top\ 1\ Accuracy = \frac{TP + TN}{test\ samples\ size}$$

Meanwhile, **precision** is used to determine the percentage of true positives against all predicted positives in the dataset. For example, when a model classified an image as a 'dress', it will have a percentage rate of $precision$ of being actually correct.

$$Precision = \frac{TP}{TP + FP}$$

Recall then is a measure of true positives predicted by the model against all predicted true positives and false negatives. This helps us to determine the ratio of true positives against misclassified true positives.

$$Recall = \frac{TP}{TP + FN}$$

The **F1-Score** is a harmonic mean of precision and recall. It is meant to symbolize both evaluation metrics equally.

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

4 Experiment

For all of our experiments, we conduct the following. Each dataset is tested on a specific CNN architecture as listed in Table 1. We measure the performance of the baseline CNN against three label noise robustness methods through four evaluation metrics: Top-1 Accuracy, Precision, Recall, and F1-Score. Finally, we discuss our findings and conclude the results of our experiment.

4.1 Datasets

We are given three noisy datasets to test our robust classifiers against: FashionMNIST0.5, Fashion-MNIST0.6, and CIFAR. There are only three possible classes for each dataset. Note that we are not working on the original datasets of FashionMNISt and CIFAR. The original datasets are estimated to be free of label noise [5], have much bigger amount of training and testing samples, and have ten levels of classes that the training data could be assigned to.

4.1.1 FashionMNIST

The FashionMNIST dataset is compromised of front look photos of various clothing items, taken from Zalando's website [8]. In total, it contains 21.000 black-and-white images of 28x28 size. FashionMNIST0.5 and FashionMNIST0.6 are the same, differing only in their noisy labels.

The three labels of FashionMNIST denote the following: 0 refers to a t-shirt/top, 1 refers to a trouser, and 2 refers to a dress.

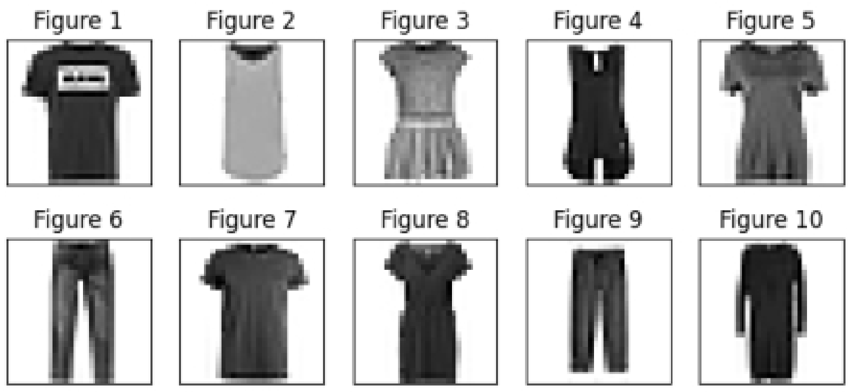

Figure 1: The first ten entries of the FashionMNIST dataset. Each image depicts a black-and-white rendition of a front look article of clothing.

4.1.2 CIFAR

The CIFAR dataset is compromised of various photos of common animals and vehicles. In the version available to us, it contains images of either a car, a cat, or an airplane. In total, it contains 18.000 colour images of 32x32 size.

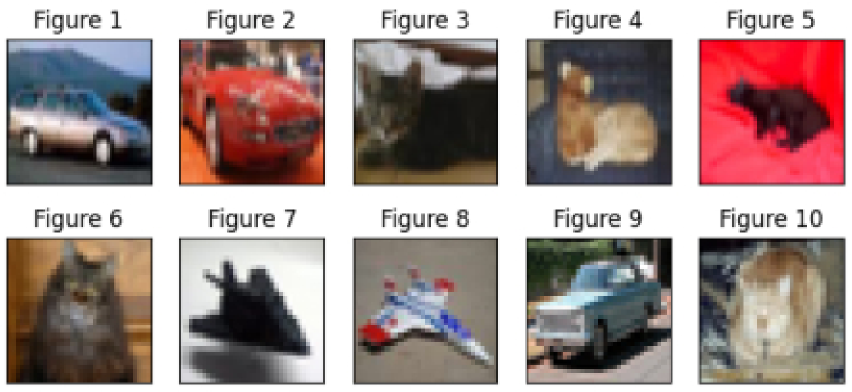

Figure 2: The first ten entries of the CIFAR dataset. Each colour image depicts either a car, cat, or an airplane.

4.2 Preprocessing

We will be dividing the value of each image from all datasets by 255. This reduces the range of values of each pixel from $[0, 255]$ to $[0, 1]$. This not only helps to reduce computational costs, but also crucial in neural networks to prevent the network stagnating at a flat domain during its neuron weighting.

4.3 Transition Matrix Estimation

We use our three CNNs to make a prediction of the test labels. The predictions are then used to estimate the anchor points in the dataset, which in turn are then used to estimate the flip rate for each label in the transition matrix. The second method entails balancing the noise rates through averaging out the flip rates for each possibility of a label being another. Using these methods, we acquired the transition matrices as listed in Table 2.

Table 2: Comparison of transition matrices

Dataset	Original	Anchor Points	Avg. Flip Rate
Fashion MNIST0.5	$\begin{bmatrix}0.5 & 0.2 & 0.3\\0.3 & 0.5 & 0.2\\0.2 & 0.3 & 0.5\end{bmatrix}$	$\begin{bmatrix}0.6426 & 0.2400 & 0.1173\\0.1674 & 0.5832 & 0.2494\\0.2347 & 0.1255 & 0.6398\end{bmatrix}$	$\begin{bmatrix}0.4531 & 0.3023 & 0.2446\\0.2218 & 0.4776 & 0.3006\\0.3061 & 0.2392 & 0.4547\end{bmatrix}$
		Sum avg. error: 0.3265	Sum avg. error: **0.2060**
Fashion MNIST0.6	$\begin{bmatrix}0.4 & 0.3 & 0.3\\0.3 & 0.4 & 0.3\\0.3 & 0.3 & 0.4\end{bmatrix}$	$\begin{bmatrix}0.4943 & 0.2579 & 0.2477\\0.2288 & 0.4855 & 0.2857\\0.2333 & 0.2358 & 0.5309\end{bmatrix}$	$\begin{bmatrix}0.3847 & 0.3038 & 0.3116\\0.2904 & 0.3878 & 0.3218\\0.3034 & 0.3107 & 0.3858\end{bmatrix}$
		Sum avg. error: 0.2071	Sum avg. error: **0.0342**
CIFAR	N/A	$\begin{bmatrix}0.5330 & 0.2974 & 0.1696\\0.2836 & 0.4756 & 0.2408\\0.3174 & 0.2150 & 0.4676\end{bmatrix}$	$\begin{bmatrix}0.3957 & 0.3157 & 0.2886\\0.3192 & 0.3718 & 0.3090\\0.3156 & 0.3144 & 0.3700\end{bmatrix}$

As we can see from Table 2, our method with the averaged out anchor points seem to have produced closer values to the original transition matrices of FashionMNIST0.5 and FashionMNIST0.6. From these results, we have evidence that suggests that our average anchor point method is a better tool for estimating noise rates.

4.4 Label Noise Methods with Known Flip Rates

In this section, we will present the result of our models using given transition matrices for Fashion-MNIST0.5 and FashionMNIST0.6 where we have known the flip rates beforehand. Please note that only noise layer adaptation and loss correction methods take transition matrix into account.

4.4.1 FashionMNIST0.5

Table 3: Mean and standard deviation report for the experiment result of FashionMNIST0.5 dataset

Mean / *Std. Dev*	**CNN**	**NL**	**LC**	**RP**
Top-1	0.9115 ± 0.01	$\mathbf{0.9465 \pm 0.00}$	0.8456 ± 0.01	0.8478 ± 0.03
Precision				
Label 0	0.8515 ± 0.03	$\mathbf{0.9302 \pm 0.01}$	0.8167 ± 0.04	0.8695 ± 0.02
Label 1	$\mathbf{0.9945 \pm 0.00}$	0.9888 ± 0.01	0.9772 ± 0.02	0.9443 ± 0.02
Label 2	0.9048 ± 0.01	$\mathbf{0.9231 \pm 0.01}$	0.7809 ± 0.04	0.8608 ± 0.01
Recall				
Label 0	$\mathbf{0.9640 \pm 0.01}$	0.9491 ± 0.01	0.9509 ± 0.02	0.7969 ± 0.06
Label 1	0.9236 ± 0.01	$\mathbf{0.9592 \pm 0.01}$	0.7898 ± 0.05	0.9382 ± 0.04
Label 2	0.8469 ± 0.04	$\mathbf{0.9312 \pm 0.01}$	0.7961 ± 0.06	0.8084 ± 0.03
F1-Score				
Label 0	0.9036 ± 0.01	$\mathbf{0.9394 \pm 0.00}$	0.8779 ± 0.02	0.8014 ± 0.06
Label 1	0.9577 ± 0.00	$\mathbf{0.9737 \pm 0.00}$	0.8723 ± 0.03	0.9391 ± 0.03
Label 2	0.8740 ± 0.02	$\mathbf{0.9269 \pm 0.00}$	0.7863 ± 0.03	0.8137 ± 0.03

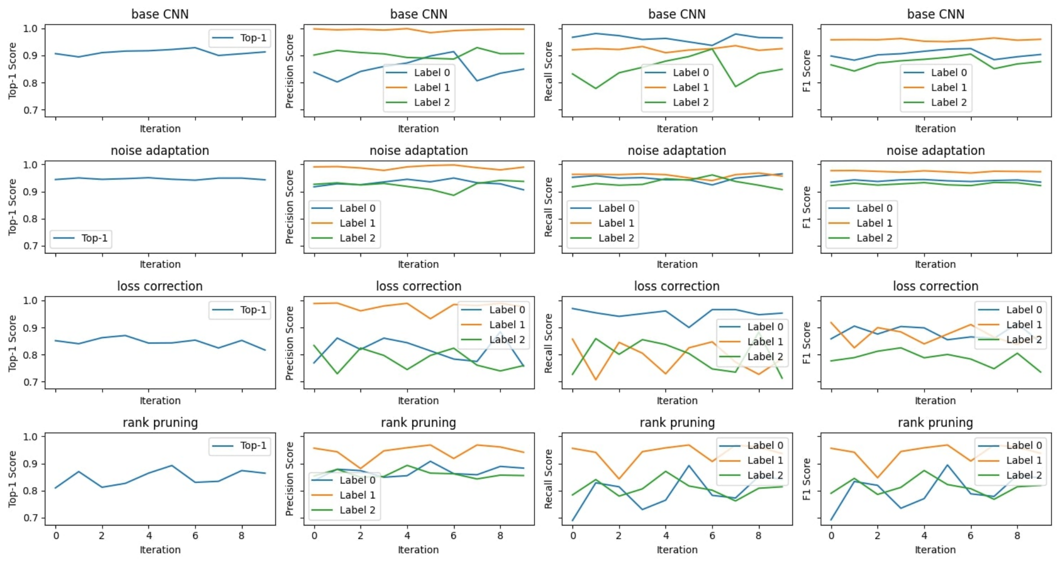

Figure 3: Graphs detailing the results of the evaluation metrics on the baseline CNN architecture and the three label noise robustness methods on the FashionMNIST0.5 dataset.

Table 4 and Fig. 3 represents the results we got for the FashionMNIST0.5 dataset. We can observe that the noise adaptation layer method performed the best and the most consistent in nearly all evaluation metrics. The baseline CNN model also performs well despite the true labels only making up 50% of the label distribution as seen in Table 2. Also, we note a near-perfect Precision score on the baseline model in its prediction of label 1 examples. On the other hand, the forward loss correction and rank pruning methods both have significantly decreased the baseline CNN's accuracy at approximately 6% decrease. Finally, we would like to point out that each model predicts label 1 consistently better compared to label 0 and label 2.

4.4.2 FashionMNIST0.6

Table 4: Mean and standard deviation report for the experiment result of FashionMNIST0.6 dataset

Mean / *Std. Dev*	**CNN**	**NL**	**LC**	**RP**
Top-1	0.8712 ± 0.01	0.8731 ± 0.01	0.8864 ± 0.01	$\mathbf{0.9062 \pm 0.01}$
Precision				
Label 0	0.8920 ± 0.03	0.8578 ± 0.04	0.8891 ± 0.02	$\mathbf{0.9128 \pm 0.02}$
Label 1	0.9523 ± 0.02	$\mathbf{0.9607 \pm 0.01}$	0.9551 ± 0.02	0.9459 ± 0.01
Label 2	0.7933 ± 0.05	0.8146 ± 0.03	0.8245 ± 0.03	$\mathbf{0.8672 \pm 0.01}$
Recall				
Label 0	0.8489 ± 0.04	0.8688 ± 0.03	0.8661 ± 0.02	$\mathbf{0.9113 \pm 0.02}$
Label 1	0.8872 ± 0.03	0.9053 ± 0.02	0.9191 ± 0.01	$\mathbf{0.9448 \pm 0.01}$
Label 2	$\mathbf{0.8775 \pm 0.05}$	0.8452 ± 0.05	0.8740 ± 0.03	0.8589 ± 0.02
F1-Score				
Label 0	0.8683 ± 0.01	0.8617 ± 0.01	0.8769 ± 0.01	$\mathbf{0.9103 \pm 0.02}$
Label 1	0.9179 ± 0.02	0.9319 ± 0.00	0.9365 ± 0.00	$\mathbf{0.9446 \pm 0.01}$
Label 2	0.8309 ± 0.02	0.8281 ± 0.01	0.8478 ± 0.01	$\mathbf{0.8606 \pm 0.01}$

We see an interesting development for the results of FashionMNIST0.6. For this dataset, Table 4 and Fig. 4 represents our findings. We see that the rank pruning method is the best method in

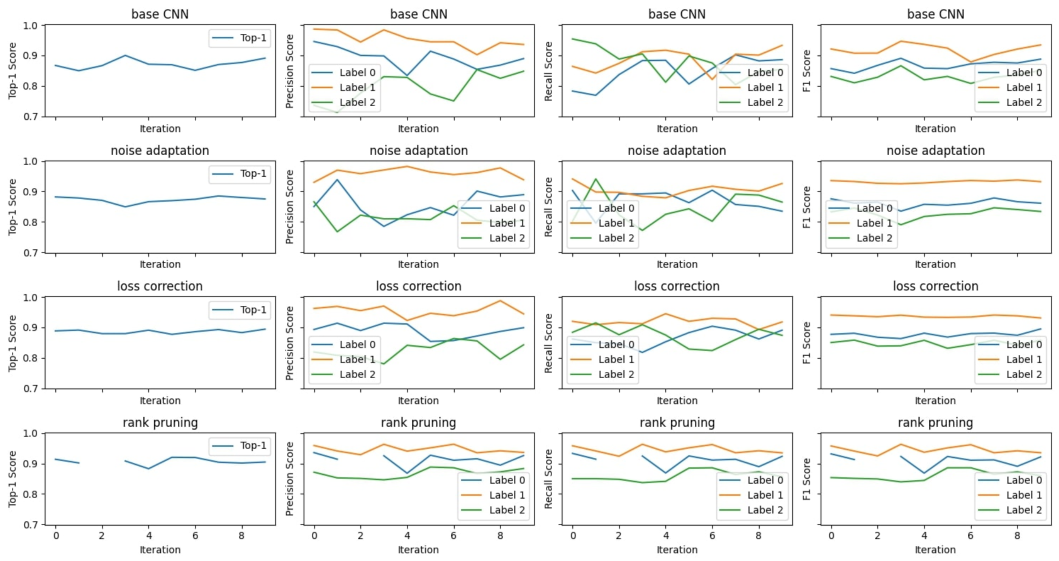

Figure 4: Graphs detailing the results of the evaluation metrics on the baseline CNN architecture and the three label noise robustness methods on the FashionMNIST0.6 dataset.

this dataset, increasing the baseline model's top-1 accuracy score by 3%. It also performs much more consistently at predicting the correct label when compared to its previous performance on the FashionMNIST0.5 dataset. The forward loss correction method's performance seem to have improved as well, this time actually improving the baseline CNN's output at a 1% increase in the same metric.

We also note that despite the performance of the rank pruning method, its values are lost in several iterations. We will discuss about this in the discussion section.

In sharp contrast to the results of the FashionMNIST0.5 dataset, the noisy adaptation layer appears to have only improved the baseline CNN architecture by a marginal value. The performance of the baseline CNN architecture itself have decreased as well compared to the FashionMNIST0.5 dataset.

4.5 Label Noise Methods with Unknown Flip Rates

In this section, for the models requiring the transition matrix, we will be using only the average anchor point transition matrix as listed in Table 2.

4.5.1 CIFAR

Table 5 and Fig. 5 represents our findings for the CIFAR dataset. Immediately, we can see a decrease of top-1 accuracy score across all models. Despite this, the rank pruning method still turns out to be the best method amongst the three label noise robustness methods, increasing the baseline CNN model's performance by 9% in its top-1 accuracy score. From Table 2, we can roughly estimate that the noise distribution of the CIFAR dataset could be similar to the FashionMNIST0.6 dataset where only 40% of the labels are correct. However, considering that it is a dataset with three channels (RGB), increasing the number of features significantly hence the lower accuracy despite similar transition matrix with FashionMNIST0.6 dataset.

In this dataset, the noise adaptation layer actually decreased the performance of the baseline CNN model by a significant amount at 9% decrease. This is notable since the baseline model also performed quite poorly at only 60% accuracy.

Table 5: Mean and standard deviation report for the experiment result of CIFAR dataset

Mean / *Std. Dev*	**CNN**	**NL**	**LC**	**RP**
Top-1	0.6049 ± 0.02	0.5124 ± 0.13	0.6562 ± 0.01	$\mathbf{0.6928 \pm 0.04}$
Precision				
Label 0	0.5979 ± 0.05	0.5550 ± 0.37	0.7186 ± 0.07	$\mathbf{0.7661 \pm 0.02}$
Label 1	$\mathbf{0.7158 \pm 0.04}$	0.4366 ± 0.21	0.6544 ± 0.05	0.6380 ± 0.06
Label 2	0.5769 ± 0.04	0.5240 ± 0.10	0.6323 ± 0.03	$\mathbf{0.6819 \pm 0.05}$
Recall				
Label 0	$\mathbf{0.7997 \pm 0.05}$	0.2859 ± 0.27	0.6609 ± 0.09	0.7596 ± 0.02
Label 1	0.3921 ± 0.12	0.4094 ± 0.23	$\mathbf{0.5787 \pm 0.09}$	0.6232 ± 0.10
Label 2	0.6229 ± 0.09	0.8418 ± 0.20	0.7289 ± 0.11	$\mathbf{0.6865 \pm 0.02}$
F1-Score				
Label 0	0.6812 ± 0.02	0.3507 ± 0.30	0.6797 ± 0.03	$\mathbf{0.7532 \pm 0.02}$
Label 1	0.4930 ± 0.09	0.3950 ± 0.20	$\mathbf{0.6072 \pm 0.04}$	0.6173 ± 0.11
Label 2	0.5937 ± 0.03	0.6205 ± 0.11	0.6703 ± 0.03	$\mathbf{0.6732 \pm 0.05}$

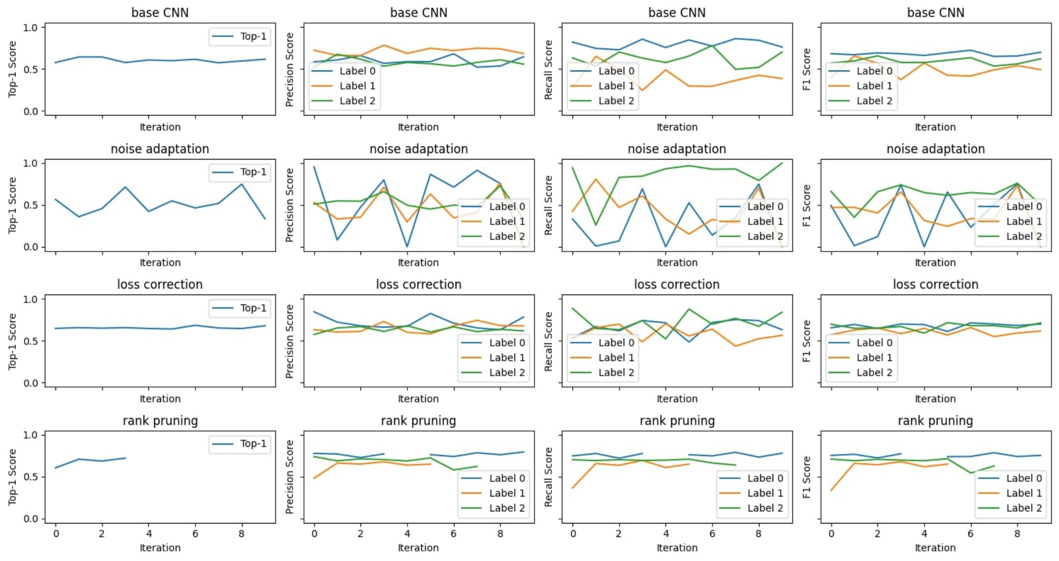

Figure 5: Graphs detailing the results of the evaluation metrics on the baseline CNN architecture and the three label noise robustness methods on the CIFAR dataset.

5 Discussion

We discuss our findings in regards to the results we acquired for each of our datasets.

We observed several trends occuring in our experiment. One that we have previously pointed out is that in the FashionMNIST0.5 and FashionMNIST0.6 datasets, each model consistently predict label 1 the most correctly when compared to label 0 and label 2. Recall that label 1 refers to a trouser while label 0 and label 2 refers to a t-shirt/top and a dress respectively. Since the structure of a trouser is starkly different compared to a top and a dress, it is reasonable that label 1 is consistently predicted very well across all models.

The second trend we observe is that the performance of our label noise robustness methods seem to fluctuate depending on the noise distribution in the label. We see in the FashionMNIST0.5 dataset where true labels are 50% of the label distribution, both the baseline CNN model and the noise adaptation layer performed very well. In the FashionMNIST0.6 dataset where true labels only make up 40% of the label distribution however, the rank pruning and forward loss correction method performed better, with the performance of the baseline model and the noise adaptation layer degrading. This trend continues onto the CIFAR dataset where true labels are roughly estimated to make up only 40% of the label distribution, rank pruning and forward loss correction methods continue to perform better, even when the baseline model only managed to reach a top-1 accuracy score of 60%.

We have a theory as to why this trend occurred. In a label distribution where the distribution of the true labels outweighs or are equal the distribution of noisy labels, we suspect that the rank pruning and forward loss correction methods may have labelled too many examples to be unreliable, causing even the correct but unreliable examples to be exempted from the training data. We theorize that this causes the models to overfit on the estimated true labels. However, this property seems to be beneficial in a label distribution where there are more noisy labels than there are true labels.

Despite the promising performance of the rank pruning method, we can observe that it has values missing in several of its iterations such as in Fig. 4 and Fig. 5. This is because the inequality of $\rho_1 + \rho_2 < 1$ is unfulfilled, thus producing no value at all in that particular iteration. The inequality is unfulfilled because of the cross-validation method when looking for the probability $P(s = 1)$ shuffled the training samples, and at some instance there might be more mislabeled samples given in the cross validation method, making the total ρ exceed 1. Note that this is a necessary condition in rank pruning, otherwise more samples would be mislabeled than labeled correctly, making the model fails altogether. We would like to investigate a workaround to accommodate this issue for our future work.

Additionally, we attempted two separate experiments to investigate the poor performance of the noise adaptation layer in the FashionMNIST0.6 dataset and dataset CIFAR dataset. The result of these experiments can be viewed in the appendix section of our Jupyter Notebook. We summarize our findings as follows. We initially hypothesized that the activation function of the baseline models might have caused inconsistency in performance as the baseline model for our FashionMNIST0.5 dataset utilised the ReLU activation function, and the ReLU activation function was also used in block [10] where this method was designed. However, after running the two experiments, there is not enough significant improvement in the performance of the noise adaptation layer. Thus, we conclude that the noise adaptation layer simply cannot cope well with high levels of noise distribution.

6 Conclusion

In this report, we compared the performance of a baseline CNN architecture and three different label noise robustness methods against three different datasets which training samples are contaminated with label noise. The performance of each model is measured through top-1 accuracy, precision, recall, and F1-score.

From our experiment, we found the following. The baseline CNN model can handle noisy labels by itself up to a certain noise rate. In the scope of our experiments, we found that its performance fall significantly at 60% noisy label distribution. The noise adaptation layer works well in improving the baseline model's performance when the amount of true labels makes up at least half of the label distribution. In training samples with higher noise rates, we found that the rank pruning method consistently performed the best amongst the label noise robustness methods.

With considerations to the scope of our experiment, we can conclude the following. If a training dataset is assumed to have a certain rate of noisy labels that does not outweigh the true labels, the noise adaptation layer is a good method to implement due to its simplicity. In contrast, if a training dataset is suspected to be heavily contaminated with label noise, the rank pruning method can be utilised to establish confident examples that the model can be reliably trained on.

We suggest the following for future related works:

- Further investigate the performance of various label noise robustness against varying levels of label noise distribution in the training set.
- Create a workaround method to rectify the inequality issue that occurs under specific conditions in the rank pruning method.

- Measure the performance difference between label noise robustness methods that utilise the transition matrix and label noise robustness methods that does not utilise the transition matrix.
- Explore and compare the robustness of machine learning algorithms under two assumptions of noisy labels: feature-dependent label noise and feature-independent label noise.

References

[1] Xinlei Chen and Abhinav Gupta. Webly Supervised Learning of Convolutional Networks. In *ICCV*, pp. 1431-1439, 2015.

[2] Adrien Halnaut, Romain Giot, Romain Bourqui, and David Auber. Compact visualization of DNN Classification performances for interpretation and improvement. In *Explainable Deep Learning AI*, Elsevier, pp. 35-54, 2003.

[3] Benoît Frénay and Michel Verleysen. Classification in Presence of Label Noise: a Survey. In *IEEE Transactions of Neural Networks and Learning Systems*, Vol. 25, No.5, 2015.

[4] Quanming Yao, Hansi Yang, Bo Han, Gang Niu, and James T. Kwok. Searching to exploit memorization effect in learning with noisy labels. In *ICML*, 2020a.

[5] Hwanjun Song, Minseok Kim, Dongmin Park, Yooju Shin, and Jae-Gil Lee. Learning from Noisy Labels with Deep Neural Networks: A Survey. In *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 34, No. 11, pp. 8135-8153, 2023.

[6] Tongliang Liu and Dacheng Tao. Classification with Noisy Labels by Importance Reweighting. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 38, No. 3, pp. 447-461, 2016.

[7] Jiacheng Cheng, Tongliang Liu, Kotagiri Ramamohanarao, and Dacheng Tao. Learning withbounded instance-and label-dependent label noise. In *ICML*, 2020.

[8] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms, arXiv e-prints, 2017. doi:10.48550/arXiv.1708.07747.

[9] Curtis G. Northcutt, Tailin Wu, Isaac L. Chuang. Learning with Confident Examples: Rank Pruning for Robust Classification with Noisy Labels, arXiv e-prints, 2017. doi:10.48550/arXiv.1705.01936.

[10] Jacob Goldberger and Ehud Ben-Reuven. Training deep neural-networks using a noise adaptation layer. In *International Conference on Learning Representations*, 2016.

[11] Zewen Li, Fan Liu, Member, Wenjie Yang, Shouheng Peng, and Jun Zhou. A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects. In *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 33, No. 12, pp. 6999-7019, 2022.

[12] Sebastian Bock and Martin Weiß. A Proof of Local Convergence for the Adam Optimizer. In *International Joint Conference on Neural Networks (IJCNN)*, pp. 1-8, 2019.

[13] Dami Choi, Christopher J. Shallue, Zachary Nado, Jaehoon Lee, Chris J. Maddison, and George E. Dahl. On Empirical Comparisons of Optimizers for Deep Learning, arXiv e-prints, 2020. doi:10.48550/arXiv.1910.05446

[14] Sainbayar Sukhbaatar, Joan Bruna, Manohar Paluri, Lubomir Bourdev, and Rob Fergus. Training Convolutional Networks with Noisy Labels, arXiv e-prints, 2015. doi:10.48550/arXiv.1406.2080

[15] Ruxin Wang, Tongliang Liu, and Dacheng Tao. Multiclass Learning With Partially Corrupted Labels. In *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 29, No. 6, pp. 2568-2580, 2018.

[16] Giorgio Patrini, Alessandro Rozza3, Aditya Krishna Menon, Richard Nock, and Lizhen Qu. Making Deep Neural Networks Robust to Label Noise: a Loss Correction Approach. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2233-2241, 2017.

[17] Sainbayar Sukhbaatar and Rob Fergus. Learning from Noisy Labels with Deep Neural Networks. In *CoRR*, 2014.

[18] Zdenek Kalal, Jiri Matas, and Krystian Mikolajczyk. P-N Learning: Bootstrapping Binary Classifiers by Structural Constraints. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 49-56, 2010.

[19] Michal Lukasik, Srinadh Bhojanapalli, Aditya Krishna Menon, and Sanjiv Kumar. Does Label Smoothing Mitigate Label Noise?, arXiv e-prints, 2020. doi:10.48550/arXiv.2003.02819

[20] Scott Reed, Honglak Lee, Dragomir Anguelov, Christian Szegedy, Dumitru Erhan, and Andrew Rabinovich. Training Deep Neural Networks on Noisy Labels with Bootstrapping, arXiv e-prints, 2014. doi:10.48550/arXiv.1412.6596