Analysis of classifiers robust to noisy labels

Abstract

We explore contemporary robust classification algorithms for overcoming class-dependant labelling noise: Forward, Importance Re-weighting and T-revision. The classifiers are trained and evaluated on class-conditional random label noise data while the final test data is clean. We demonstrate methods for estimating the transition matrix in order to obtain better classifier performance when working with noisy data. We apply deep learning to three data-sets and derive an end-to-end analysis with unknown noise on the CIFAR data-set from scratch. The effectiveness and robustness of the classifiers are analysed, and we compare and contrast the results of each experiment are using top-1 accuracy as our criterion.

1 Introduction

The advances in deep learning techniques, particularly in applications such as image classification, has put even greater importance on the accuracy of labels as we forge ahead in the big data era. The need for human input and intervention to identify, categorise or otherwise add context to data via labels is just one way that can also give rise to error and inaccuracies which we loosely term noise [9]. A variety of techniques have been proposed to tackle this problem and at the same time bring a number of assumptions of practical importance to machine learning practitioners [1].

Label noise is a significant obstacle when working with the massive are varied data-sets typically used in training advanced machine learning models, such as deep neural networks. Research has also shown that precision of the learned classifiers can be profoundly influenced by label noise [3, 9, 13]. Noisy labelled data-set training induces output loss because deep neural networks (DNNs) will easily over-fit the noise labels [4]. For a basic cause, the dilemma is pervasive: manual technical labelling of each case on a wide scale is not possible, and researchers often turn to inexpensive yet incomplete surrogates [2, 7].

We set out to examine robust multi-class classification methods under a number of scenarios in which we leverage noisy labelled data-sets to derive insights and knowledge that applies to the underlying clean labelled data distribution. We develop two classifiers using known class-dependent, asymmetric flip rates which indicate the likelihood a member of each class has had its label changed. We then develop a procedure to estimate unknown flip rates applied to the CIFAR dataset. Finally, we demonstrate the effectiveness of our methods using top-1 accuracy from samples drawn from the true clean distribution.

2 Previous Work

Methods for label-noise learning can be generally divided into two categories: statistically consistent/inconsistent, and risk consistent classifiers. The first category looks to limit the impact noisy data through typically heuristic means i.e. expert selection of samples or label correction [11]. Methods belonging to the risk category assume label noise is randomly conditioned on the true labels [8] and look to minimise (1) through a process of *loss-correction* where $Q(f_\theta(x_i)) = p(\tilde{y}|f_\theta(x_i))$. Q can be formulated with a noise transition matrix T so that $Q(f_\theta(x_i)) = Tf_\theta(x_i)$ where each element of the matrix represents the transition probability of $\tilde{y}$ noisy to y true, $T_{ij} = p(\tilde{y} = j|y = i)$ [1].

$$\hat{R}_{l,D}(f) = \frac{1}{N}\sum_i^N l(Q(f_\theta(x_i)), \hat{y}_i) \tag{1}$$

The noise transition matrix, denoting the probability of clean labels flipping into noisy labels, plays a central role in constructing statistically accurate classifiers in label-noise research. Current theories have demonstrated that by exploiting anchor points, the transition matrix can be trained (i.e. data points that almost definitely belong to a certain class). Given an instance x, if $P(Y = i|X = x) \approx 1$ and otherwise $P(Y = k|X = x) = 0$ where $k \neq i$ we have (12). However, the transition matrix will be improperly learned where there are no anchor points, and these formerly stable classifiers will greatly degenerate [11].

$$P(\overline{Y} = j|X = x) = \sum_1^C T_{kj}P(Y = k|X = x) = T_{ij} \tag{2}$$

Sample Importance Weighting allows training to be made more effective by assigning weights to instances according to their estimated noisiness level. These methods look to minimise the empirical risk using a form (3) whereby a dynamic function $\beta(X, Y)$ determines the instance dependant weight [1]. It is expected that a correctly labelled example has a large $\beta(X, Y)$ and contributes more to the risk, while an incorrectly labelled example has a smaller value and will contribute less [10]. However, these methods can also sometimes become biased towards a certain subset of data [1]. In Figure 1, we can see a number of 'flipped' class labels on the right-hand side. The estimated distributions of D and $\hat{D}$ can be used to suppress the influence of these incorrect labels [5, 12, 10, 11].

$$\hat{R}_{l,D}(f) = \frac{1}{N}\sum_{i=1}^N \beta(x_i, y_i)l(f_\theta(x_i), \hat{y}_i) \tag{3}$$
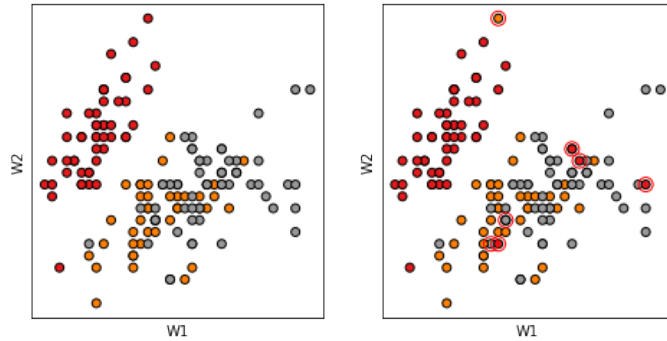

Figure 1: Illustration of label noise. Left: True labels drawn from D, Right: Noisy labels drawn from $\hat{D}$. Red circles indicate corrupted or 'flipped' labels

3 Label Noise Methods with Known Flip Rates

We examine robust methods for multi-class image classification using already known transition matrices and compare each method using top-1 accuracy. We examine the "Forward" method and the Importance of Re-weighting method. We introduce each formulation and the theoretical basis for robustness as well as details we employ to leverage these methods. Experiments and results for each method are discussed at length in Section 6.

3.1 Forward Learning

The Forward Learning method incorporates the known T matrix into the learning procedure with the use of a noise adaption layer. Using the known noisy labelled dataset, we could simply train a network to best match the noisy labels $\hat{P}(\bar{Y}|X)$. Instead, we explicitly introduce the dependency on T, which allows us to compare the noisy labels to averaged noisy predictions corrupted by T [8, 6]. We can use the cross-entropy loss function (4,5,6) to illustrate this process, which consequently allows us to approximate $P(Y|X)$. The noise layer serves as a normal linear layer but has no bias and its weights will range between 0 and 1 because the represent conditional probabilities T_{ij} [8].

$$\ell\left(e^i, \hat{p}(\boldsymbol{y} \mid \boldsymbol{x})\right) = -\log \hat{p}\left(\tilde{\boldsymbol{y}} = e^i \mid \boldsymbol{x}\right) \tag{4}$$

$$= -\log \sum_{j=1}^{c} p\left(\tilde{\boldsymbol{y}} = e^i \mid \boldsymbol{y} = e^j\right) \hat{p}\left(\boldsymbol{y} = e^j \mid \boldsymbol{x}\right) \tag{5}$$

$$= -\log \sum_{j=1}^{c} T_{ji} \hat{p}\left(\boldsymbol{y} = e^j \mid \boldsymbol{x}\right) \tag{6}$$

3.2 Formulation

Provided the noise matrix T is non singular, a proper composite loss (for example, cross entropy) has the forward loss correction defined as (7) [6].

$$\ell_\psi(\boldsymbol{h}(\boldsymbol{x})) = \ell\left(T^\top \boldsymbol{\psi}^{-1}(\boldsymbol{h}(\boldsymbol{x}))\right) \tag{7}$$

The minimizer of the corrected loss under a noisy distribution is the same as the minimizer of the initial loss under that clean distribution (8) [6]. Since T is already known, we can minimise w.r.t the clean data distribution on the basis $\hat{P}(Y|X)$ approximates $P(Y|X)$. We make use of a custom neural network architecture in Figure 7 to iteratively learn $\hat{P}(Y|X)$ using the classifier in Figure 2.

$$\operatorname*{argmin}_{h} \mathbb{E}_{\boldsymbol{x},\tilde{y}} \overrightarrow{\ell_\psi}(\boldsymbol{y}, \boldsymbol{h}(\boldsymbol{x})) = \operatorname*{argmin}_{\boldsymbol{h}} \mathbb{E}_{\boldsymbol{x},\boldsymbol{y}} \ell_\psi(\boldsymbol{y}, \boldsymbol{h}(\boldsymbol{x})) \tag{8}$$
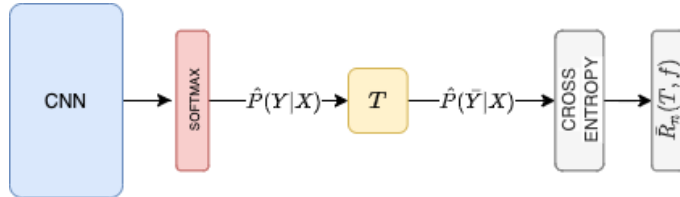

Figure 2: Forward classifier.

3.3 Importance Re-weighting

Sample importance weighting looks to adjust the effects of likely noisy labels using weights. The Importance Re-weighting method uses principles from domain adaptation which takes knowledge of the source domain (the noisy distribution) $\bar{D}$ to improve model performance in a target domain (the true or 'clean' distribution) D. Here the method uses the joint probability of (X, Y) under the two distributions D and $\bar{D}$ [10].

$$\beta(X, \hat{Y}) = \frac{P_D(X, Y)}{P_{\bar{D}}(X, \hat{Y})} = \frac{P_D(Y|X)}{P_{\bar{D}}(\hat{Y}|X)} \tag{9}$$

With the true distribution D unknown, we use the known transition matrix T under the assumption $P_D(X) = P_{\hat{D}}(X)$ to empirically estimate the true distribution D.

3.3.1 Formulation

The Importance Re-weighting technique is used to rewrite the expected risk w.r.t clean data given noise is independent of instances in a multi-class setting which also avoids the computationally intensive inverse of the transition matrix. The risk-consistent estimator (10) is derived given $f(X) = argmax_j \in \{1, \ldots, C\} g_j(X)$ where $g_j(X)$ is an estimate for $P(Y = j|X)$ and w denotes the loss function is weighted [11].

$$\bar{R}_{n,w}(T, f) = \frac{1}{n} \sum \frac{g_{\bar{Y}_i}(X_i)}{(T^\intercal g)_{\bar{Y}_i}(X_i)} \ell(f(X_i), \bar{Y}_i) \tag{10}$$

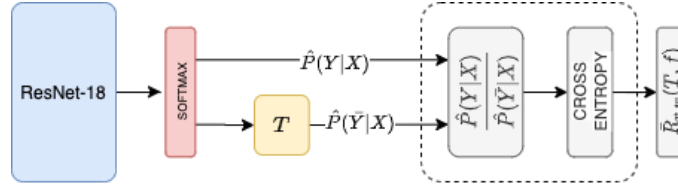$$\beta(X, Y) = \frac{g_{\bar{Y}_i}(X_i)}{(T^\intercal g)_{\bar{Y}_i}(X_i)} \tag{11}$$

Figure 3: Importance Re-weighting classifier.

Using the provided T as fixed, we use the softmax function to approximate $g(x) = \hat{P}(Y|X = x) \approx P(Y|X = x)$ and $T^\intercal g(x) = \hat{P}(\bar{Y}|X = x) \approx P(\bar{Y}|X = x)$. A ResNet-18 (5) deep learning architecture is used to learn $\beta(X, Y)$ which together with the cross-entropy loss function trains the classifier on the noisy dataset. Predictions on the clean dataset are produced by the classifier having learned $P(Y|X = x)$.

4 Noise Rate Estimation Method

We leverage the noisy dataset to learn the noisy class posteriors $P(\tilde{Y}|X)$ iteratively during the training of a deep learning classifier. The entire data set is used to identify instances which exhibit high noisy class posterior probabilities. We are able to estimate the fixed class-conditional noise empirically by estimating anchor points using learned information within the noisy dataset. The T matrix (12) is then constructed comprising of a C rows and columns, where C represents the number of classes. Each entry indicates the likelihood of a noisy label given the true label. We developed a series of experiments to confirm the accuracy of this method in Section 6.4.2.

$$T = \begin{bmatrix} P(\tilde{Y} = 0|Y = 0), P(\tilde{Y} = 0|Y = 1), P(\tilde{Y} = 0|Y = 2) \\ P(\tilde{Y} = 1|Y = 0), P(\tilde{Y} = 1|Y = 1), P(\tilde{Y} = 1|Y = 2) \\ P(\tilde{Y} = 2|Y = 0), P(\tilde{Y} = 2|Y = 1), P(\tilde{Y} = 2|Y = 2) \end{bmatrix} \tag{12}$$

4.1 Formulation

We implemented the ResNet-18 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 4 shows the architecture of the basic blocks and the Figure 5 presents the entire ResNet-18 architecture. Additionally, the softmax function is utilised to estimate the class of each instance.
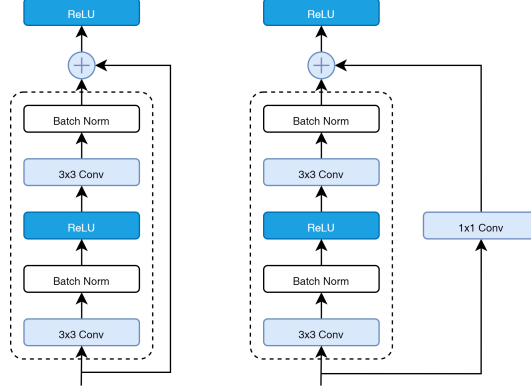

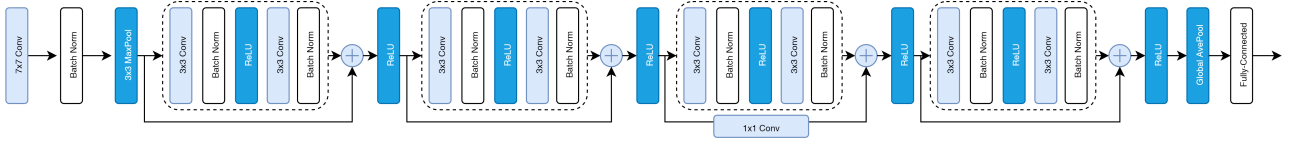Figure 4: Basic ResNet Block without and with $1x1$ convolution.

Figure 5: ResNet-18 Architecture.

5 Label Noise Methods with Unknown Flip Rates

5.1 T-Revision

Using already established transition matrices, we analyse rigorous methods for multi-class image classification and compare each approach using top-1 accuracy. The T-revision approach suggested by [11] is investigated. In general, we initially configure the transition matrix by leveraging samples that are closer to the anchor points, including those with posterior probabilities of high approximate noisy class. By adding a slack vector afterwards, we alter the initial matrix, which will then be trained and validated along with the classifier by using only the noisy data.

The suggested method of T-revision works because by minimising the risk-consistent estimator, which is asymptotically equivalent to the anticipated risk w.r.t. clean data, we learn ΔT [11]. On the noisy validation set, the learned idle variable may also be validated, i.e. to verify whether $\widehat{P}(\overline{Y}|X = x)$ matches the validation set. The principle of this technique is close to that of the strategy of cross-validation.

5.1.1 Formulation

The T-Revision approach is divided into two main stages as observed in Figure 6. We initially use the noisy training data to learn $\hat{T}$ by learning the noisy class posterior probabilities and approximating its initial state (4). In a similar fashion as the Importance Re-weighting method (10) described previously, we now use T and incrementally adjust it by $\triangle T$ throughout the training process whereas previously it was fixed.
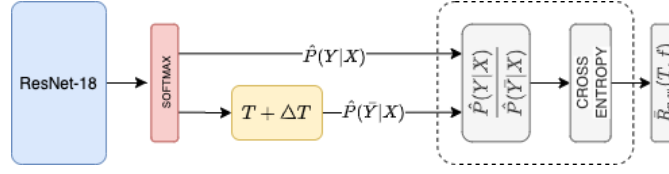
5

Figure 6: T-Revision Re-weighting classifier.

6 Experiment and Results

In this study, we create a series of experiments to prove the implementation of the previous methods and analyse the robustness to label noise of the different classifiers. Additionally, we validate the effectiveness of our transition matrix estimator using the true transition matrix provided.

The datasets used to conduct the experiments are FashionMNIST0.5, FashionMNIST0.6 and Cifar10. The FashionMNIST0.5 and FashionMNIST0.6 dataset contain 18000 images for training and validation. The images are in grayscale and the shape of each sample is $(28x28)$. On the other hand, the Cifar10 dataset contains 15000 colour images for training and validation, the shape of each sample is $(32x32x3)$. The classifiers will evaluate on the test set provided for each dataset, which each one contains 3000 samples.

6.1 Pre-processing

We apply no further pre-processing aside from a simple pixel value transformation whereby we convert 0-255 pixel values to values between 0-1 which make up the input features.

6.2 Metrics

To compare the performance and robustness of the different classifiers, we use the following metric:

- **Top-1 Accuracy:** The output of each classifier will be evaluated using the top-1 precision metric.

$$\text{top-1 accuracy} = \frac{\text{number of correctly classified examples}}{\text{total number of test examples}} * 100$$

On the other hand, to evaluate the effectiveness of our transition matrix estimator we use:

- **Sum Average:** This metric allows us to measure the differences between the estimated transition matrix and the true transition matrix provided.

$$\text{sum average} = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} |\widehat{T}_{ij}|}{\sum_{i=1}^{m} \sum_{j=1}^{n} |T_{ij}|}$$

where T denotes the true transition matrix and $\widehat{T}$ denotes the estimated transition matrix.

To correctly perform the evaluation of these experiments, we train each classifier 10 times with the different training and validation sets generated by random sampling. Then we extract both the mean and the standard deviation of the accuracy of the test.

6.3 Hardware

The code has been developed using Python 3x and Jupyter Notebook and we have taken advantage of the computational capacity provided by Google Colab. However, a computer with the following specifications could easily run these experiments:

Model	MSI GE63VR 7RF
Processor	Intel Core i7 7700HQ 2.80GHz
RAM	16gb
Graphics	GTX 1070 8GB

6.4 Transition Matrix Estimator

In this section, we estimate the transition matrix for the Cifar10 dataset using the transition matrix estimator proposed in the previous section. Additionally, we evaluate the effectiveness of our estimator using the provided transition matrices for the FashionMNIST0.5 and FashionMNIST0.6 datasets.

6.4.1 Estimate Transition Matrix - Cifar10

As we mentioned previously, we used the ResNet architecture to learn the noisy class posterior $P(\widetilde{Y}|X)$ and empirically estimate the anchor points on the noisy dataset and use these anchor points to estimate the transition matrix. The Figure 4 shows the architecture of the basic blocks and the Figure 5 presents the entire ResNet-18 architecture.

The parameters used to train the neural network is contained in the Table 1. In this case and to properly learn the noisy class posterior we trained the neural network during 10 iterations using the Stochastic Gradient Descent (SGD) and Cross-Entropy as the loss function.

Iterations	10
Optimizer	SGD[1]
Learning Rate (lr)	0.001
Momentum	0.9
Loss Function	Cross-Entropy

[1] Average Stochastic Gradient Descent.

Table 1: Configuration used to train the ResNet-18 architecture to estimate the transition matrix.

The transition matrix $\widehat{T}$ estimated using the noisy class posterior for the Cifar10 dataset is presented below:

$$\widehat{T} = \begin{bmatrix} 0.439 & 0.301 & 0.259 \\ 0.283 & 0.467 & 0.249 \\ 0.278 & 0.290 & 0.431 \end{bmatrix} \tag{13}$$

6.4.2 Validating the effectiveness of our estimator

To analyse the effectiveness of our estimator we use the provided transition matrices for the FashionMNIST0.5 and FashionMNIST0.6 datasets. We trained the ResNet-18 model using these datasets to learn the noisy class posterior and then generate the estimated transition matrix. Finally, we evaluate the difference between the transition matrix estimated by our estimator and the true matrix using the sum average metric (section 6.2). The results are shown in the Table 2.

We can observe how the estimated matrices $\widehat{T}$ using our estimator are very similar to the true matrices T provided. For the FashionMNIST0.5 dataset, the sum average error between $\widehat{T}$ and T is around 0.158. On the other hand, the error for the estimated transition matrix using the FashionMNIST0.6 dataset is around 0.09. In this case, the estimator manages to approximate the matrix very well and the small error may be due to the number of decimals that the estimated matrix contains. We can conclude that our estimator produces a very satisfactory result and we can consider the estimated transition matrix for the Cifar10 dataset accurate.

Dataset	T	$\widehat{T}$	Sum Average Error
Cifar10	-	$\begin{bmatrix} 0.439 & 0.301 & 0.259 \\ 0.283 & 0.467 & 0.249 \\ 0.278 & 0.290 & 0.431 \end{bmatrix}$	-
FashionMNIST0.5	$\begin{bmatrix} 0.5 & 0.2 & 0.3 \\ 0.3 & 0.5 & 0.2 \\ 0.2 & 0.3 & 0.5 \end{bmatrix}$	$\begin{bmatrix} 0.545 & 0.224 & 0.229 \\ 0.231 & 0.488 & 0.280 \\ 0.285 & 0.213 & 0.501 \end{bmatrix}$	**0.158**
FashionMNIST0.6	$\begin{bmatrix} 0.4 & 0.3 & 0.3 \\ 0.3 & 0.4 & 0.3 \\ 0.3 & 0.3 & 0.4 \end{bmatrix}$	$\begin{bmatrix} 0.475 & 0.250 & 0.274 \\ 0.273 & 0.433 & 0.292 \\ 0.289 & 0.281 & 0.429 \end{bmatrix}$	**0.09**

Table 2: Comparison between the provided true transition matrix and the estimated transitions matrix using our estimator.

6.5 Robustness to Noisy Labels

In this section, we provide the results for the analysis of the robustness to noisy labels of the classifiers mentioned previously.

6.5.1 Forward Learning

As we mentioned previously, we implemented a Convolutional Neural Network with the Forward Learning method. The architecture of this neural network is presented in Figure 7.
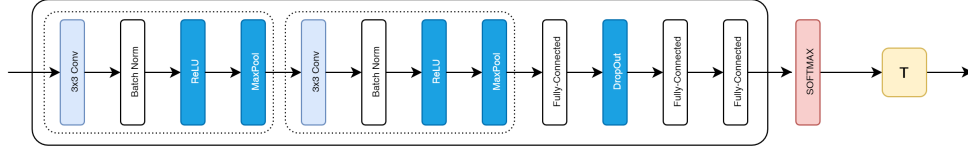

Figure 7: Convolutional Neural Network implemented with Forward Learning.

In Figure 8 we present the accuracy on the test set using Forward Learning for the different datasets. The top panel of the figure contains the accuracy for each training time and the bottom panel shows the mean and standard deviation value.
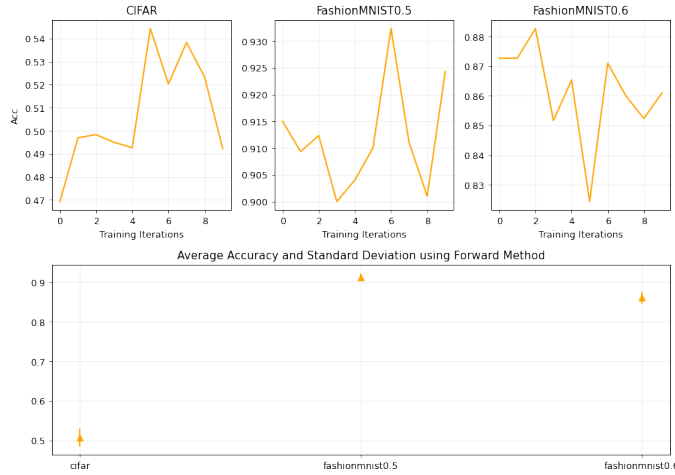

Figure 8: Average Accuracy and Standard Deviation using Forward Method.

It can be observed how the performance for the FashionMNIST0.5 and FashionMNIST0.6 is pretty similar, around 0.87. On the other hand, the accuracy of the Cifar10 dataset is around 0.5.

6.5.2 Importance Re-weighting

In this experiment, we trained the ResNet model defined previously and we used importance re-weighting to adjust the effects of noisy labels. The architecture of the neural network is presented in Figure 5.

Figure 9 presents the accuracy on the test set using the importance re-weighting method. As we mentioned previously, the top panel shows the evolution of the accuracy for each training time and the bottom panel contains the average accuracy and the standard deviation for each dataset. The dataset Cifar10 has experienced an increase in accuracy from 0.50 to 0.60. However, the accuracy for the FashionMNIST0.5 and FashionMNIST0.6 remains practically the same, around 0.90 and 0.85 respectively.
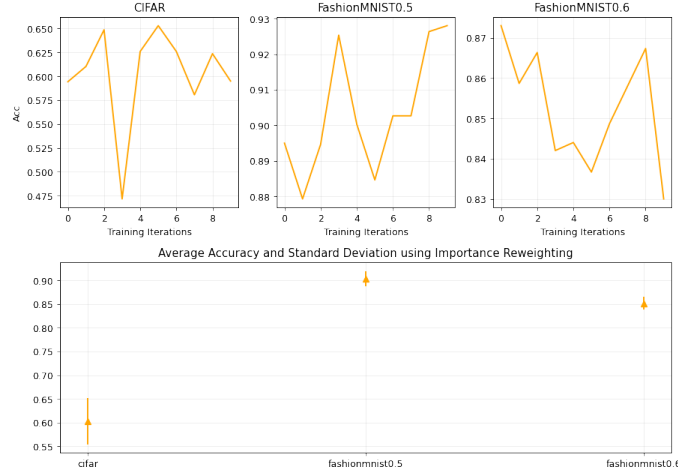

Figure 9: Average Accuracy and Standard Deviation using Importance Re-weighting.

6.5.3 T-Revision

As we mentioned previously, we implemented T-Revision using the ResNet model defined in the last section.
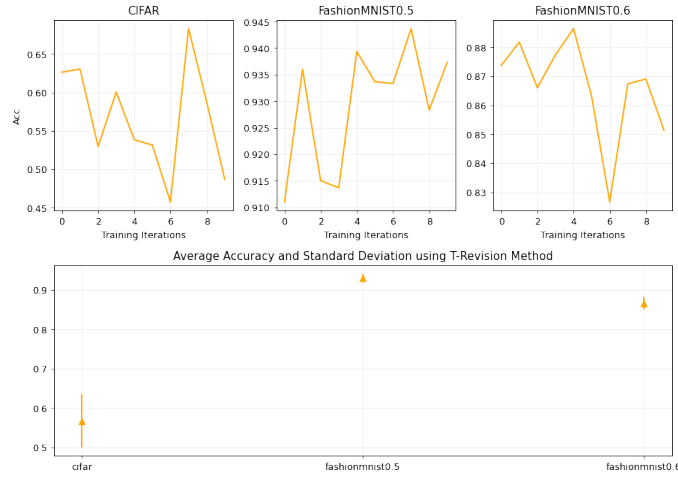

Figure 10: Average Accuracy and Standard Deviation using T-Revision.

The results presented in Figure 10 show how the accuracy for the dataset Cifar10 experienced a decrease from 0.60 to 0.58 with respect to the importance re-weighting method. Nevertheless, the

accuracy for the FashionMNIST0.5 and FashionMNIST0.6 dataset has slightly increased from 0.90 to 0.92 and from 0.85 to 0.88 respectively.

As T-Revision allows us to learn ΔT, in the Table 3 we present the value of ΔT, $\widehat{T} + \Delta T$ for each dataset.

Dataset	$\widehat{T}$			ΔT			$(\widehat{T} + \Delta T)$		
Cifar10[1]	0.439	0.301	0.259	0.0332	0.0366	0.0286	0.4726	0.3386	0.2872
	0.283	0.467	0.249	0.0416	0.0449	0.0462	0.3246	0.5122	0.2960
	0.278	0.290	0.431	0.0322	0.0508	0.0372	0.3111	0.3409	0.4683
FashionMNIST0.5	0.5	0.2	0.3	0.0279	0.0216	0.0400	0.5279	0.2216	0.3400
	0.3	0.5	0.2	0.0243	0.0219	0.0228	0.3243	0.5219	0.2228
	0.2	0.3	0.5	0.0307	0.0331	0.0282	0.2307	0.3331	0.5282
FashionMNIST0.6	0.4	0.3	0.3	0.0482	0.0340	0.0452	0.4482	0.3340	0.3452
	0.3	0.4	0.3	0.0389	0.0447	0.0420	0.3389	0.4447	0.3420
	0.3	0.3	0.4	0.0463	0.0338	0.0434	0.3463	0.3338	0.4434

[1] For the Cifar10 dataset, the $\widehat{T}$ was estimated using the estimator implemented in the previous section.

Table 3: Transition matrices with the correction ΔT learned using T-Revision.

6.5.4 Comparison

This section contains a comparison between the results previously presented and the baseline model for each dataset. The baseline model refers to the same model (ResNet or CNN) trained without applying any technique to correct the effect of the noisy labels. We analyse if the methods implemented to improve the performance of the baseline model.

Cifar10

Figure 11 shows the average accuracy and standard deviation for the Forward, Importance Re-weighting and T-Revision methods. As we can see, the accuracy using the Forward method is the lowest of all, around 0.51. On the other hand, it seems that the performance of the baseline model is better than the Forward method, around 0.54. However, the accuracy obtained using Importance Re-weighting and T-Revision is slightly higher with respect to the baseline model, around 0.60 and 0.565 respectively.
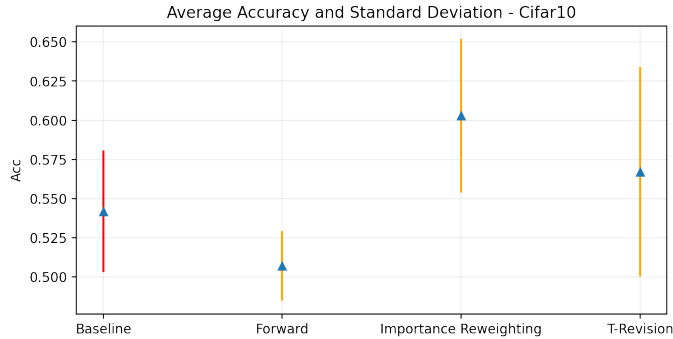

Figure 11: Comparison of the average accuracy and standard deviation for different methods - Cifar10

FashionMNIST0.5

As we can observe in the Figure 12, the accuracy obtained for FashionMNIST0.5 is pretty higher and it seems that we achieved a very good result just using the baseline model, around 0.91. Additionally, the accuracy using the Forward and Importance Re-weighting method is very similar to the one obtained using the baseline model, around 0.91 and 0.905 respectively. On the other hand, using T-Revision we observe an increase in accuracy up to 0.93.
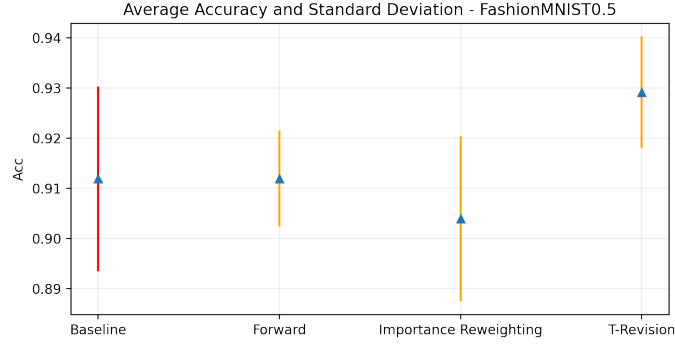
Figure 12: Comparison of the average accuracy and standard deviation for different methods - FashionMNIST0.5

As we mentioned before, the performance using the baseline model, T-Revision and Importance Re-weighting method is very similar and we can therefore draw the conclusion that the transition matrix provided is not very accurate and thus the result of these methods is highly similar to the baseline model. However, the T-Revision method allows us to slightly increase the accuracy, this is because this method allows us to update the transition matrix during the training and thus obtain a more accurate matrix.

FashionMNIST0.6
Figure 13 shows that in contrast to the results obtained in the other datasets, we obtain higher accuracy with respect to the baseline model in all the methods used. In this case, the baseline model obtained an accuracy of around 0.82. Nevertheless, the accuracy achieved with Forward and Importance Re-weighting method is very similar, around 0.86. On the other hand, T-Revision is the method which obtained the highest accuracy, about 0.865.
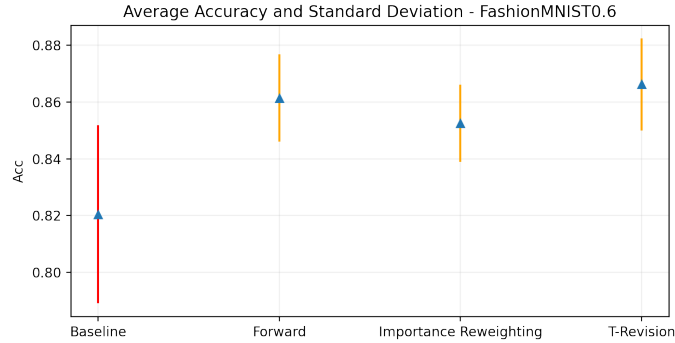

Figure 13: Comparison of the average accuracy and standard deviation for different methods - FashionMNIST0.6

In this case and observing the results obtained, we can conclude that the transition matrix provided is accurate and that the transition matrix for the cifar10 dataset has been accurately estimated. As we mentioned before, the T-Revision method allows us to obtain a higher accuracy due to update the transition matrix and this enables us to get a more accurate result.

7 Conclusion

In this project, we presented and implemented the formulation of three different methods to make the models robust to noisy labels. Additionally, we implemented a transition matrix estimator to estimate the transition matrix for the Cifar10 dataset. We analysed the performance of these methods and the effectiveness of our estimator.

It could be observed that the estimator implemented to estimate the transition matrix produces a highly accurate result. We evaluated its effectiveness using the transition matrices provided for FashionMNIST0.5 and FashionMNIST0.6 dataset and the errors obtained from comparing the matrices obtained using our estimator and the true matrices are very small so we concluded that our estimator produces a very accurate result.

On the other hand, we analysed the behaviour of our classifiers to noisy labels when the Forward, Importance Re-weighting and T-Revision methods are applied. Firstly, we concluded that Importance Re-weighting is the method which achieved the highest accuracy for the Cifar10 dataset, followed closely by T-Revision. In parallel, we found that the transition matrix provided for the Fashion-MNIST0.5 dataset may not be very accurate because the Forward and Importance Re-weighting methods obtain a very similar result to the baseline model. However, the T-Revision achieved a higher accuracy thus, we concluded that the transition matrix for this dataset may not be accurate and we should use T-Revision or our estimator to obtain a better approximation of the transition matrix for this case. Finally, we observed how the models trained using the FashionMNIST0.6 dataset experienced an improvement using the Forward, Importance Re-weighting or T-Revision methods.

In the future, we can extend this study in the following aspects:

- Study if it is possible to estimate the transition matrices at the same time as we train the classifiers.
- Extend this work to more complex datasets, such as ImageNet and analyse its performance.
- Estimate the transition matrix for the FashionMNIST0.5 dataset and analyse if the provided transition matrix has been well estimated.

References

[1] Görkem Algan and Ilkay Ulusoy. *Image Classification with Deep Learning in the Presence of Noisy Labels: A Survey*. 2020. arXiv: `1912.05170 [cs.LG]`.

[2] Rob Fergus et al. "Learning object categories from internet image searches". In: *Proceedings of the IEEE* 98.8 (2010), pp. 1453–1466.

[3] Benoıt Frénay and Michel Verleysen. "Classification in the presence of label noise: a survey". In: *IEEE transactions on neural networks and learning systems* 25.5 (2013), pp. 845–869.

[4] Junnan Li et al. "Learning to learn from noisy labeled data". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 5051–5059.

[5] Tongliang Liu and Dacheng Tao. "Classification with noisy labels by importance reweighting". In: *IEEE Transactions on pattern analysis and machine intelligence* 38.3 (2015), pp. 447–461.

[6] Giorgio Patrini et al. "Making deep neural networks robust to label noise: A loss correction approach". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 1944–1952.

[7] Florian Schroff, Antonio Criminisi, and Andrew Zisserman. "Harvesting image databases from the web". In: *IEEE transactions on pattern analysis and machine intelligence* 33.4 (2010), pp. 754–766.

[8] Sainbayar Sukhbaatar and Rob Fergus. "Learning from noisy labels with deep neural networks". In: *arXiv preprint arXiv:1406.2080* 2.3 (2014), p. 4.

[9] Sainbayar Sukhbaatar et al. "Training convolutional networks with noisy labels". In: *arXiv preprint arXiv:1406.2080* (2014).

[10] Ruxin Wang, Tongliang Liu, and Dacheng Tao. "Multiclass learning with partially corrupted labels". In: *IEEE transactions on neural networks and learning systems* 29.6 (2017), pp. 2568–2580.

[11] Xiaobo Xia et al. "Are Anchor Points Really Indispensable in Label-Noise Learning?" In: *NeurIPS*. 2019.

[12] Tong Xiao et al. "Learning from massive noisy labeled data for image classification". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 2691–2699.

[13] Chiyuan Zhang et al. "Understanding deep learning requires rethinking generalization". In: *arXiv preprint arXiv:1611.03530* (2016).

8 Appendix

The code of this work is provided in the Jupyter Notebook called *Analysis_of_classifiers_robust_to_noisy_labels.ipynb* and the experiments were executed using Python 3x in Google Colab.

To execute the code you just need to go through the notebook and executing all cells. The first section called *Load and preprocessing data* contains the function to load and preprocess the images. In this section, you should add the path to the folder with the datasets, in case you use Google Colab you need to add the datasets to your Drive account and mount your Drive folder into the notebook (figure 14).

```
available_datasets = {'cifar': '/content/drive/My Drive/comp5328-assignment-2/data/CIFAR.npz',
                      'fashionmnist0.5': '/content/drive/My Drive/comp5328-assignment-2/data/FashionMNIST0.5.npz',
                      'fashionmnist0.6': '/content/drive/My Drive/comp5328-assignment-2/data/FashionMNIST0.6.npz'}
```

Figure 14: Provide a path to the target dataset.

Section *Utils Functions* contains functions that are used in the implementation of our algorithms. The PyTorch function to load the dataset, generate predictions as well as the function to estimate the transition matrix are contained in this section. Furthermore, the section called *Classifiers robust to label noise* contains the implementation of the different proposed methods. Finally, section *Experiments* contains the experiments we have executed to analyse the effectiveness of our estimator and the performance of the classifiers to noisy labels using the different methods.