ROBUST IMAGE CLASSIFICATION IN THE PRESENCE OF LABEL NOISE: AN EMPIRICAL STUDY OF PREPROCESSING, TRANSITION MATRIX ESTIMATION, AND LOSS CORRECTION TECHNIQUES

Andrea Bosia
520600843
abos6574@uni.sydney.edu.au

Dylan Leon
480378174
dleo4585@uni.sydney.edu.au

Yuchen Li
490236424
yuli5818@uni.sydney.edu.au

ABSTRACT

This report details the development of classifiers with enhanced noise robustness, an essential feature for improving the performance of machine learning models when faced with label noise in training datasets. The robustness of two neural network architectures, ResNet and GoogLeNet, is investigated over three datasets: Fashion MNIST 0.5, Fashion MNIST 0.6, and CIFAR. Our study utilizes forward and backward loss-correction techniques along with an important reweighting approach, employing transition matrices to train unbiased estimators for the clean data distribution from noisy datasets. For CIFAR, we estimate the transition matrix using anchor points due to the absence of a predefined matrix, while for Fashion MNIST, we use the provided matrices. Our result indicates the superiority of forward correction in improving accuracy and F1 scores. The report concludes with insights into the comparative performance of the correction techniques and their implications for future noise-robust model development.

1 Introduction

Deep learning has revolutionised the field of computer vision, with neural networks becoming the standard approach for this diverse and challenging breadth of tasks [1]. Image classification stands as a cornerstone task in this field, with applications ranging from medical diagnosis to autonomous driving. However, the success of models in this domain is dependent on the availability of large, accurately labelled datasets, which are often expensive and time-consuming to obtain. In practice, manual expert-labelling of each instance is not feasible, so the use of imperfect labelling processes such as crowdsourcing non-expert labels or using heuristics to infer labels is common [2]. While these methods are cost-effective, they generally introduce noise into the dataset, which can have a significant adverse effect on the performance of machine learning models.

Noise, defined as anything that obscures the relationship between an instance's features and its class, is generally classified in two categories: feature (or attribute) noise and class noise [3], [4]. Feature noise affects the observed values of the instance's feature, while class noise alters the observed label assigned to an instance. This report focuses on class noise, as it has been shown to be more harmful than feature noise in classification tasks [5]. Specifically, we focus on class-conditional label noise, where the noise is dependent on the instance's true class (and independent of the instance's features) [3]. This noise can stem from various sources, such as human annotation errors, ambiguous instances, or biases in the data collection process. The presence of this type of noise in a model's training data can significantly deteriorate test performance as the model overfits to the noisy labels. Thus, creating classifiers that maintain their performance robustness despite the presence of noise is crucial for the development of resilient and dependable machine learning systems [6], [7].

While various families of methods have been proposed to address the problem of class-conditional label noise, all approaches can be separated into two categories: noise model based and noise model free methods. The scope of this report extends only to the former. Noise model based methods leverage the assumed prior knowledge of the underlying noise structure to either extract noise-free information contained in the dataset or reform the dataset by correcting noisy labels [8]. The noise structure is typically represented by a transition matrix that models the probability, or flip rate, of a label being flipped into another label.

This report investigates the challenge of designing classifiers that are robust to class-conditional label noise. We evaluate the performance of two distinguished CNN architectures, ResNet18 and GoogLeNet, when trained with noisy labels. To address the noise, we implement and compare three algorithmic strategies that leverage transition matrices: forward loss correction, backward loss correction, and importance reweighting. Moreover, we investigate the estimation of transition matrices through anchor points – a set of data points with high confidence in their labels, which serve as a reference for correcting other labels [9]. The efficacy of this approach is relevant when the actual noise rates are unknown, a common scenario in real-world datasets.

By undertaking this assignment, we aim to contribute valuable insights and methodologies that enhance the field's understanding of noise robustness in classification tasks, driving the development of more resilient and accurate machine-learning models.

This paper is organised as follows. Section 2 provides an overview of the literature on image classification in the presence of label noise, highlighting the use of transition matrices as an approach to modelling and correcting for label noise. Section 12 details the methodology adopted in our experiments, including the use of transition matrices and the implementation of various loss correction techniques. The experimental setup

is presented in Section 13, followed by the results and discussion in Section 14. Finally, Section 15 concludes the paper and outlines potential avenues for future research.

2 Previous Work

2.1 Robust Learning Through Loss Adjustment

The literature on deep learning in the presence of label noise has seen various methods proposed to address the challenges posed by incorrect annotations. This report focuses on the group of methods that attempt to account for noise through loss adjustment during training. These methods reduce the negative impact of noisy labels by adjusting the loss of all training examples before each update of the deep neural network (DNN). By adjusting the loss, these methods aim to build an unbiased estimator of the loss function, such that under expected label noise the corrected loss equals the original one computed on clean data [2]. Song et al. [1] present four subcategories of methods in this group.

1. ***Loss Correction:*** This approach modifies the loss for each example by multiplying the estimated label transition probability by the output of the DNN for every instance. Forward and Backward learning, two of the methods implemented in this paper, are loss correction algorithms. Their theory and implementation is detailed in Section 12.
2. ***Loss Reweighting:*** Loss reweighting, a development of importance reweighting [10], weighs the loss of training instances by their probability that their label is contaminated with noise. Samples with smaller weights do not contribute as much to the training loss, and so do not significantly affect learning.

   The adjusted update rule for learning model parameters Θ on the mini-batch $\mathcal{B}_t$ is given by:

   $$\Theta_{t+1} = \Theta_t - \eta \nabla \left(\frac{1}{|\mathcal{B}_t|} \sum_{(x,\tilde{y}) \in \mathcal{B}_t} \overbrace{w(x,\tilde{y}) \ell(f(x;\Theta_t),\tilde{y})}^{\text{Reweighted Loss}} \right) \tag{1}$$

   where $w(x,\tilde{y})$ is the weight of an example x with its noisy label $\tilde{y}$ [1].
3. ***Label Refurbishment:*** This approach "replaces" noisy labels with a refurbished label obtained by a convex combination of the noisy label and the network's prediction. The model then backpropagates the loss for the refurbished label instead of the noisy one.

   $$y_{\text{Refurb.}} = \alpha \tilde{y} + (1-\alpha)\hat{y} \tag{2}$$

   where $\hat{y}$ is the current prediction of the network, $\tilde{y}$ is the noisy label, and $\alpha \in [0,1]$ is the label confidence of $\tilde{y}$.
4. ***Meta Learning:*** Meta learning has become an important topic in the machine learning community as an improved method based on *learning to learn*, thereby being data scenario-agnostic and automating the application of the previously defined approaches [11].

Of these categories, we implement three specific algorithms for dealing with the noise. Specifically, we implement forward and backward loss correction algorithms, as well as loss reweighting.

Importantly, these methods differ from other approaches in that they decouple the learning of the noise distribution structure from the training of the model [1]. As such, not only do these approaches require that the noise structure be known (or estimated) before training, but their effectiveness is highly dependent on the quality of the assumed noise distribution. This is a key disadvantage, as it is often very difficult to get a perfect representation of underlying noise distribution in real-world datasets.

2.2 Transition Matrix

The distribution of noise in a dataset is typically represented by the square transition matrix $T \in [0,1]^{c \times c}$, where the probability of one label i being flipped to another j is given by T_{ij} [9]. Noise correction methods that utilise the transition matrix infer the clean class posterior probability from the noisy class posterior probability and the transition matrix.

2.2.1 Transition Matrix Estimation

As previously mentioned, the effectiveness of robust learning algorithms that employ loss adjustment is highly dependent on the quality of the assumed underlying noise structure. Considering that retrieving a perfect noise distribution from noisy data is unlikely for real-world scenarios, much research has been conducted into the estimation of transition matrices from contaminated data.

Natarjan et al. [7] present a cross-validation approach for estimating the noise probabilities in a binary classification setting, however this method is not tractable for the multi-class scenario. A frequently used alternative relies on the presence of *anchor points*, defined as instances that belong to a true label class almost surely. After training a model to find these anchor points, the transition matrix is then estimated using posterior noisy label distributions of those points [12].

2.3 GoogLeNet and ResNet

Our implementation of the GoogLeNet model is fundamentally based on the innovative architecture known as Inception, which was introduced by [13]. The Inception architecture is notable for its strategic use of computing resources within the network, allowing for increased depth and width without a proportional rise in computational cost. This is achieved through the Inception module, which organises network computation across different scales and uses dimension-reduction techniques to maintain computational efficiency.

A few key points of the Inception architecture include:

- Multi-Scale Processing: By incorporating filters of various sizes (1x1, 3x3, and 5x5), the network can capture information at multiple scales.
- Dimension Reduction: The use of 1x1 convolutions before larger convolutions reduces dimensionality, thus controlling computational requirements.
- Depth and Width: The architecture allows for deeper (22 layers in GoogLeNet) and wider networks without a significant increase in computational demand.

The GoogLeNet model, which utilises this architecture, has demonstrated significant improvements in classification and detection tasks, setting new benchmarks in the ImageNet Large-Scale Visual Recognition Challenge 2014 (ILSVRC14). The principles guiding the design of the Inception architecture, such as the Hebbian principle and

multi-scale processing, have been empirically validated, making it a cornerstone for our model's development.

Another method employed ResNet as the backbone and estimated a label transition matrix to deal with symmetric label noise in CIFAR10 and CIFAR100 datasets. The details of the method were not fully accessible due to restrictions, but the paper indicated that transition matrix estimation played a crucial role in handling noisy labels within a ResNet-34 architecture [14].

Each of these methods, employing transition matrices in conjunction with neural network architectures such as LeNet and ResNet, aims at mitigating the effects of label noise on image classification tasks. They show the potential of transition matrices in providing a structured approach to understanding and correcting label noise, which is crucial for the success of machine learning models in real-world, noisy settings.

3 Problem Setup

The following notation is adopted:

Observation:	$X \in \mathcal{X} \subseteq \mathbb{R}^d$
Clean but unobservable label:	$Y \in \mathcal{Y} = \{-1, +1\}$
Observable but noisy label:	$\tilde{Y} \in \mathcal{Y}$
Clean distribution:	$D(X, Y)$
Noisy distribution:	$D_\rho(\mathbf{X}, \tilde{Y})$

Label noise is a hurdle that may occur with classification tasks. Specifically, samples affected by noise end up being labelled with an incorrect class. Ultimately, this results in a dataset for which the only observable distribution $D_\rho(X, \tilde{Y})$ is the noisy one, as opposed to the clean one $D(X, Y)$. Where ρ represents the flip rate, hence the probability of a label to be flipped into a different one given an observation X. In the case of class conditional label noise the flip rate is defined as the posterior probability $\rho_Y(\mathbf{X}) = P(\tilde{Y}|Y, \mathbf{X})$ [15].

In a canonical classification setting the goal is to minimise the expected 0-1 risk. defined as,

$$R_D(f) = P(\text{sign}(f(X)) \neq Y) = \mathbb{E}_{(X,Y)\sim D}[1(\text{sign}(f(X)) \neq Y)] \tag{3}$$

However, given that drawbacks inherent to the 0-1 function, the objective shifts toward the minimisation of the expectation of the surrogate loss function (and more precisely, its empirical formulation):

$$\textbf{Expected L-Risk:} \qquad R_{D,L}(f) = \mathbb{E}_{(X,Y)\sim D}\left[L(f(X), Y)\right] \tag{4}$$

$$\textbf{Empirical Risk:} \qquad R_{D,L,n}(f) = \frac{1}{n}\sum_{i=1}^{n} L(f(X_i), Y_i) \tag{5}$$

The problem with addressing class-conditional label noise, is that during training we are only exposed to $D_\rho(X, \tilde{Y})$, and not the clean distribution $D(X, Y)$. Hence, the goal now becomes to find a risk function for which the following holds [2]:

$$\arg\min_{f\in\mathcal{F}} R_{D,L}(f) = \arg\min_{f\in\mathcal{F}} R_{D_\rho,\tilde{L}}(f) \tag{6}$$

4 Methodology

We approach the task of designing a noise-robust classifier by comparing the effectiveness of three loss adjustment approaches: loss reweighting, forward loss correction, and backward loss correction.

4.1 Loss Reweighting

To address the previously formulated problem (Eq. 30) a possibility is to approach it as a domain adaptation task.

Given that we are dealing with two different distributions between the noisy training set and the clean test set, finding a robust loss function for the target domain (i.e. the clean dataset) can indeed be seen as a domain adaptation problem. Adopting techniques of importance reweighing we get [15]:

$$\begin{aligned}
R_{e,D}(f) &= R[D,f,\ell] = \mathbb{E}_{(X,Y)\sim D}[\ell(f(X),Y)] \\
&= \mathbb{E}_{(X,\hat{Y})\sim D_\rho}\left[\frac{P_D(X,Y)}{P_{D_\rho}(X,\hat{Y})}\ell(f(X),\hat{Y})\right] \\
&= R\left[D_\rho, f, \frac{P_D(X,Y)}{P_{D_\rho}(X,\hat{Y})}\ell(f(X),\hat{Y})\right] \\
&= R\left[D_\rho, f, \beta(X,\hat{Y})\ell(f(X),\hat{Y})\right] \\
&= R_{\beta,D_\rho}(f),
\end{aligned}$$

Where, in the presence of label noise,

$$P_D(X) = P_D\rho(X)$$

Hence,

$$\beta(X,\hat{Y}) = \frac{P_D(X,Y)}{P_{D_\rho}(X,\hat{Y})} = \frac{P_D(Y|X)P_D(X)}{P_{D_\rho}(\hat{Y}|X)P_{D_\rho}(X)} = \frac{P_D(Y|X)}{P_{D_\rho}(\hat{Y}|X)} \tag{7}$$

This result demonstrates how, by means of the joint probabilities $\beta(X,\tilde{Y})$, we can effectively adjust the loss function computed on the noisy distribution in order to learn a noise robust classifier for the clean target domain.

Such a classifier can be learnt solving the following minimisation problem:

$$\arg\min R_{\beta,D_\rho}(f) = \arg\min_{f\in\mathcal{F}} \frac{1}{n}\sum_{i=1}^{n}\beta(X_i,\hat{Y}_i)\ell(f(X_i),\hat{Y}_i) \tag{8}$$

In other terms the empirical risk minimisation problem is reformulated so that the loss function takes into account how likely it is for a label to be flipped in D_ρ. Indeed, as previously shown, β represent the ratio between the clean and noisy posterior. Hence, ultimately, the more a label is affected by noise, the higher the weight assigned to it.

Figure 1 clarifies how the loss reweighting method is implemented when training a Deep Neural Network for such a classification task.

5 Problem Setup

The following notation is adopted:

Observation:	$X \in \mathcal{X} \subseteq \mathbb{R}^d$
Clean but unobservable label:	$Y \in \mathcal{Y} = \{-1, +1\}$
Observable but noisy label:	$\tilde{Y} \in \mathcal{Y}$
Clean distribution:	$D(X, Y)$
Noisy distribution:	$D_\rho(\mathbf{X}, \tilde{Y})$

Label noise is a hurdle that may occur with classification tasks. Specifically, samples affected by noise end up being labelled with an incorrect class. Ultimately, this results in a dataset for which the only observable distribution $D_\rho(X, \tilde{Y})$ is the noisy one, as opposed to the clean one $D(X, Y)$. Where ρ represents the flip rate, hence the probability of a label to be flipped into a different one given an observation X. In the case of class conditional label noise the flip rate is defined as the posterior probability $\rho_Y(\mathbf{X}) = P(\tilde{Y}|Y, \mathbf{X})$ [15].

In a canonical classification setting the goal is to minimise the expected 0-1 risk. defined as,

$$R_D(f) = P(\text{sign}(f(X)) \neq Y) = \mathbb{E}_{(X,Y)\sim D}[1(\text{sign}(f(X)) \neq Y)] \tag{9}$$

However, given that drawbacks inherent to the 0-1 function, the objective shifts toward the minimisation of the expectation of the surrogate loss function (and more precisely, its empirical formulation):

$$\textbf{Expected L-Risk:} \qquad R_{D,L}(f) = \mathbb{E}_{(X,Y)\sim D}\left[L(f(X), Y)\right] \tag{10}$$

$$\textbf{Empirical Risk:} \qquad R_{D,L,n}(f) = \frac{1}{n}\sum_{i=1}^{n} L(f(X_i), Y_i) \tag{11}$$

The problem with addressing class-conditional label noise, is that during training we are only exposed to $D_\rho(X, \tilde{Y})$, and not the clean distribution $D(X, Y)$. Hence, the goal now becomes to find a risk function for which the following holds [2]:

$$\arg\min_{f\in\mathcal{F}} R_{D,L}(f) = \arg\min_{f\in\mathcal{F}} R_{D_\rho,\tilde{L}}(f) \tag{12}$$

6 Methodology

We approach the task of designing a noise-robust classifier by comparing the effectiveness of three loss adjustment approaches: loss reweighting, forward loss correction, and backward loss correction.

6.1 Loss Reweighting

To address the previously formulated problem (Eq. 30) a possibility is to approach it as a domain adaptation task.

Given that we are dealing with two different distributions between the noisy training set and the clean test set, finding a robust loss function for the target domain (i.e. the clean dataset) can indeed be seen as a domain adaptation problem. Adopting techniques of importance reweighing we get [15]:

$$\begin{aligned} R_{e,D}(f) &= R[D, f, \ell] = \mathbb{E}_{(X,Y)\sim D}[\ell(f(X), Y)] \\ &= \mathbb{E}_{(X,\hat{Y})\sim D_\rho}\left[\frac{P_D(X,Y)}{P_{D_\rho}(X,\hat{Y})}\ell(f(X),\hat{Y})\right] \\ &= R\left[D_\rho, f, \frac{P_D(X,Y)}{P_{D_\rho}(X,\hat{Y})}\ell(f(X),\hat{Y})\right] \\ &= R\left[D_\rho, f, \beta(X,\hat{Y})\ell(f(X),\hat{Y})\right] \\ &= R_{\beta,D_\rho}(f), \end{aligned}$$

Where, in the presence of label noise,

$$P_D(X) = P_D\rho(X)$$

Hence,

$$\beta(X,\hat{Y}) = \frac{P_D(X,Y)}{P_{D_\rho}(X,\hat{Y})} = \frac{P_D(Y|X)P_D(X)}{P_{D_\rho}(\hat{Y}|X)P_{D_\rho}(X)} = \frac{P_D(Y|X)}{P_{D_\rho}(\hat{Y}|X)} \tag{13}$$

This result demonstrates how, by means of the joint probabilities $\beta(X,\tilde{Y})$, we can effectively adjust the loss function computed on the noisy distribution in order to learn a noise robust classifier for the clean target domain.

Such a classifier can be learnt solving the following minimisation problem:

$$\arg\min R_{\beta,D_\rho}(f) = \arg\min_{f\in\mathcal{F}} \frac{1}{n}\sum_{i=1}^{n}\beta(X_i,\hat{Y}_i)\ell(f(X_i),\hat{Y}_i) \tag{14}$$

In other terms the empirical risk minimisation problem is reformulated so that the loss function takes into account how likely it is for a label to be flipped in D_ρ. Indeed, as

previously shown, β represent the ratio between the clean and noisy posterior. Hence, ultimately, the more a label is affected by noise, the higher the weight assigned to it.

Figure 1 clarifies how the loss reweighting method is implemented when training a Deep Neural Network for such a classification task.

7 Problem Setup

The following notation is adopted:

Observation:	$X \in \mathcal{X} \subseteq \mathbb{R}^d$
Clean but unobservable label:	$Y \in \mathcal{Y} = \{-1, +1\}$
Observable but noisy label:	$\tilde{Y} \in \mathcal{Y}$
Clean distribution:	$D(X, Y)$
Noisy distribution:	$D_\rho(\mathbf{X}, \tilde{Y})$

Label noise is a hurdle that may occur with classification tasks. Specifically, samples affected by noise end up being labelled with an incorrect class. Ultimately, this results in a dataset for which the only observable distribution $D_\rho(X, \tilde{Y})$ is the noisy one, as opposed to the clean one $D(X, Y)$. Where ρ represents the flip rate, hence the probability of a label to be flipped into a different one given an observation X. In the case of class conditional label noise the flip rate is defined as the posterior probability $\rho_Y(\mathbf{X}) = P(\tilde{Y}|Y, \mathbf{X})$ [15].

In a canonical classification setting the goal is to minimise the expected 0-1 risk. defined as,

$$R_D(f) = P(\text{sign}(f(X)) \neq Y) = \mathbb{E}_{(X,Y)\sim D}[1(\text{sign}(f(X)) \neq Y)] \tag{15}$$

However, given that drawbacks inherent to the 0-1 function, the objective shifts toward the minimisation of the expectation of the surrogate loss function (and more precisely, its empirical formulation):

$$\textbf{Expected L-Risk:} \qquad R_{D,L}(f) = \mathbb{E}_{(X,Y)\sim D}\left[L(f(X), Y)\right] \tag{16}$$

$$\textbf{Empirical Risk:} \qquad R_{D,L,n}(f) = \frac{1}{n}\sum_{i=1}^{n} L(f(X_i), Y_i) \tag{17}$$

The problem with addressing class-conditional label noise, is that during training we are only exposed to $D_\rho(X, \tilde{Y})$, and not the clean distribution $D(X, Y)$. Hence, the goal now becomes to find a risk function for which the following holds [2]:

$$\arg\min_{f\in\mathcal{F}} R_{D,L}(f) = \arg\min_{f\in\mathcal{F}} R_{D_\rho,\tilde{L}}(f) \tag{18}$$

8 Methodology

We approach the task of designing a noise-robust classifier by comparing the effectiveness of three loss adjustment approaches: loss reweighting, forward loss correction, and backward loss correction.

8.1 Loss Reweighting

To address the previously formulated problem (Eq. 30) a possibility is to approach it as a domain adaptation task.

Given that we are dealing with two different distributions between the noisy training set and the clean test set, finding a robust loss function for the target domain (i.e. the clean dataset) can indeed be seen as a domain adaptation problem. Adopting techniques of importance reweighing we get [15]:

$$\begin{aligned} R_{e,D}(f) &= R[D, f, \ell] = \mathbb{E}_{(X,Y)\sim D}[\ell(f(X), Y)] \\ &= \mathbb{E}_{(X,\hat{Y})\sim D_\rho}\left[\frac{P_D(X,Y)}{P_{D_\rho}(X,\hat{Y})}\ell(f(X),\hat{Y})\right] \\ &= R\left[D_\rho, f, \frac{P_D(X,Y)}{P_{D_\rho}(X,\hat{Y})}\ell(f(X),\hat{Y})\right] \\ &= R\left[D_\rho, f, \beta(X,\hat{Y})\ell(f(X),\hat{Y})\right] \\ &= R_{\beta,D_\rho}(f), \end{aligned}$$

Where, in the presence of label noise,

$$P_D(X) = P_D\rho(X)$$

Hence,

$$\beta(X,\hat{Y}) = \frac{P_D(X,Y)}{P_{D_\rho}(X,\hat{Y})} = \frac{P_D(Y|X)P_D(X)}{P_{D_\rho}(\hat{Y}|X)P_{D_\rho}(X)} = \frac{P_D(Y|X)}{P_{D_\rho}(\hat{Y}|X)} \tag{19}$$

This result demonstrates how, by means of the joint probabilities $\beta(X,\tilde{Y})$, we can effectively adjust the loss function computed on the noisy distribution in order to learn a noise robust classifier for the clean target domain.

Such a classifier can be learnt solving the following minimisation problem:

$$\arg\min R_{\beta,D_\rho}(f) = \arg\min_{f\in\mathcal{F}} \frac{1}{n}\sum_{i=1}^{n}\beta(X_i,\hat{Y}_i)\ell(f(X_i),\hat{Y}_i) \tag{20}$$

In other terms the empirical risk minimisation problem is reformulated so that the loss function takes into account how likely it is for a label to be flipped in D_ρ. Indeed, as previously shown, β represent the ratio between the clean and noisy posterior. Hence, ultimately, the more a label is affected by noise, the higher the weight assigned to it.

Figure 1 clarifies how the loss reweighting method is implemented when training a Deep Neural Network for such a classification task.

9 Problem Setup

The following notation is adopted:

Observation:	$X \in \mathcal{X} \subseteq \mathbb{R}^d$
Clean but unobservable label:	$Y \in \mathcal{Y} = \{-1, +1\}$
Observable but noisy label:	$\tilde{Y} \in \mathcal{Y}$
Clean distribution:	$D(X, Y)$
Noisy distribution:	$D_\rho(\mathbf{X}, \tilde{Y})$

Label noise is a hurdle that may occur with classification tasks. Specifically, samples affected by noise end up being labelled with an incorrect class. Ultimately, this results in a dataset for which the only observable distribution $D_\rho(X, \tilde{Y})$ is the noisy one, as opposed to the clean one $D(X, Y)$. Where ρ represents the flip rate, hence the probability of a label to be flipped into a different one given an observation X. In the case of class conditional label noise the flip rate is defined as the posterior probability $\rho_Y(\mathbf{X}) = P(\tilde{Y}|Y, \mathbf{X})$ [15].

In a canonical classification setting the goal is to minimise the expected 0-1 risk. defined as,

$$R_D(f) = P(\text{sign}(f(X)) \neq Y) = \mathbb{E}_{(X,Y)\sim D}[1(\text{sign}(f(X)) \neq Y)] \tag{21}$$

However, given that drawbacks inherent to the 0-1 function, the objective shifts toward the minimisation of the expectation of the surrogate loss function (and more precisely, its empirical formulation):

$$\textbf{Expected L-Risk:} \qquad R_{D,L}(f) = \mathbb{E}_{(X,Y)\sim D}\left[L(f(X), Y)\right] \tag{22}$$

$$\textbf{Empirical Risk:} \qquad R_{D,L,n}(f) = \frac{1}{n}\sum_{i=1}^{n} L(f(X_i), Y_i) \tag{23}$$

The problem with addressing class-conditional label noise, is that during training we are only exposed to $D_\rho(X, \tilde{Y})$, and not the clean distribution $D(X, Y)$. Hence, the goal now becomes to find a risk function for which the following holds [2]:

$$\arg\min_{f\in\mathcal{F}} R_{D,L}(f) = \arg\min_{f\in\mathcal{F}} R_{D_\rho,\tilde{L}}(f) \tag{24}$$

10 Methodology

We approach the task of designing a noise-robust classifier by comparing the effectiveness of three loss adjustment approaches: loss reweighting, forward loss correction, and backward loss correction.

10.1 Loss Reweighting

To address the previously formulated problem (Eq. 30) a possibility is to approach it as a domain adaptation task.

Given that we are dealing with two different distributions between the noisy training set and the clean test set, finding a robust loss function for the target domain (i.e. the clean dataset) can indeed be seen as a domain adaptation problem. Adopting techniques of importance reweighing we get [15]:

$$\begin{aligned} R_{e,D}(f) &= R[D, f, \ell] = \mathbb{E}_{(X,Y)\sim D}[\ell(f(X), Y)] \\ &= \mathbb{E}_{(X,\hat{Y})\sim D_\rho}\left[\frac{P_D(X,Y)}{P_{D_\rho}(X,\hat{Y})}\ell(f(X),\hat{Y})\right] \\ &= R\left[D_\rho, f, \frac{P_D(X,Y)}{P_{D_\rho}(X,\hat{Y})}\ell(f(X),\hat{Y})\right] \\ &= R\left[D_\rho, f, \beta(X,\hat{Y})\ell(f(X),\hat{Y})\right] \\ &= R_{\beta,D_\rho}(f), \end{aligned}$$

Where, in the presence of label noise,

$$P_D(X) = P_D\rho(X)$$

Hence,

$$\beta(X,\hat{Y}) = \frac{P_D(X,Y)}{P_{D_\rho}(X,\hat{Y})} = \frac{P_D(Y|X)P_D(X)}{P_{D_\rho}(\hat{Y}|X)P_{D_\rho}(X)} = \frac{P_D(Y|X)}{P_{D_\rho}(\hat{Y}|X)} \tag{25}$$

This result demonstrates how, by means of the joint probabilities $\beta(X,\tilde{Y})$, we can effectively adjust the loss function computed on the noisy distribution in order to learn a noise robust classifier for the clean target domain.

Such a classifier can be learnt solving the following minimisation problem:

$$\arg\min R_{\beta,D_\rho}(f) = \arg\min_{f\in\mathcal{F}} \frac{1}{n}\sum_{i=1}^{n}\beta(X_i,\hat{Y}_i)\ell(f(X_i),\hat{Y}_i) \tag{26}$$

In other terms the empirical risk minimisation problem is reformulated so that the loss function takes into account how likely it is for a label to be flipped in D_ρ. Indeed, as previously shown, β represent the ratio between the clean and noisy posterior. Hence, ultimately, the more a label is affected by noise, the higher the weight assigned to it.

Figure 1 clarifies how the loss reweighting method is implemented when training a Deep Neural Network for such a classification task.

11 Problem Setup

The following notation is adopted:

Observation:	$X \in \mathcal{X} \subseteq \mathbb{R}^d$
Clean but unobservable label:	$Y \in \mathcal{Y} = \{-1, +1\}$
Observable but noisy label:	$\tilde{Y} \in \mathcal{Y}$
Clean distribution:	$D(X, Y)$
Noisy distribution:	$D_\rho(\mathbf{X}, \tilde{Y})$

Label noise is a hurdle that may occur with classification tasks. Specifically, samples affected by noise end up being labelled with an incorrect class. Ultimately, this results in a dataset for which the only observable distribution $D_\rho(X, \tilde{Y})$ is the noisy one, as opposed to the clean one $D(X, Y)$. Where ρ represents the flip rate, hence the probability of a label to be flipped into a different one given an observation X. In the case of class conditional label noise the flip rate is defined as the posterior probability $\rho_Y(\mathbf{X}) = P(\tilde{Y}|Y, \mathbf{X})$ [15].

In a canonical classification setting the goal is to minimise the expected 0-1 risk. defined as,

$$R_D(f) = P(\text{sign}(f(X)) \neq Y) = \mathbb{E}_{(X,Y)\sim D}[1(\text{sign}(f(X)) \neq Y)] \tag{27}$$

However, given that drawbacks inherent to the 0-1 function, the objective shifts toward the minimisation of the expectation of the surrogate loss function (and more precisely, its empirical formulation):

$$\textbf{Expected L-Risk:} \qquad R_{D,L}(f) = \mathbb{E}_{(X,Y)\sim D}\left[L(f(X), Y)\right] \tag{28}$$

$$\textbf{Empirical Risk:} \qquad R_{D,L,n}(f) = \frac{1}{n}\sum_{i=1}^{n} L(f(X_i), Y_i) \tag{29}$$

The problem with addressing class-conditional label noise, is that during training we are only exposed to $D_\rho(X, \tilde{Y})$, and not the clean distribution $D(X, Y)$. Hence, the goal now becomes to find a risk function for which the following holds [2]:

$$\arg\min_{f\in\mathcal{F}} R_{D,L}(f) = \arg\min_{f\in\mathcal{F}} R_{D_\rho,\tilde{L}}(f) \tag{30}$$

12 Methodology

We approach the task of designing a noise-robust classifier by comparing the effectiveness of three loss adjustment approaches: loss reweighting, forward loss correction, and backward loss correction.

12.1 Loss Reweighting

To address the previously formulated problem (Eq. 30) a possibility is to approach it as a domain adaptation task.

Given that we are dealing with two different distributions between the noisy training set and the clean test set, finding a robust loss function for the target domain (i.e. the clean dataset) can indeed be seen as a domain adaptation problem. Adopting techniques of importance reweighing we get [15]:

$$\begin{aligned} R_{e,D}(f) &= R[D, f, \ell] = \mathbb{E}_{(X,Y)\sim D}[\ell(f(X), Y)] \\ &= \mathbb{E}_{(X,\hat{Y})\sim D_\rho}\left[\frac{P_D(X,Y)}{P_{D_\rho}(X,\hat{Y})}\ell(f(X),\hat{Y})\right] \\ &= R\left[D_\rho, f, \frac{P_D(X,Y)}{P_{D_\rho}(X,\hat{Y})}\ell(f(X),\hat{Y})\right] \\ &= R\left[D_\rho, f, \beta(X,\hat{Y})\ell(f(X),\hat{Y})\right] \\ &= R_{\beta,D_\rho}(f), \end{aligned}$$

Where, in the presence of label noise,

$$P_D(X) = P_D\rho(X)$$

Hence,

$$\beta(X,\hat{Y}) = \frac{P_D(X,Y)}{P_{D_\rho}(X,\hat{Y})} = \frac{P_D(Y|X)P_D(X)}{P_{D_\rho}(\hat{Y}|X)P_{D_\rho}(X)} = \frac{P_D(Y|X)}{P_{D_\rho}(\hat{Y}|X)} \tag{31}$$

This result demonstrates how, by means of the joint probabilities $\beta(X,\tilde{Y})$, we can effectively adjust the loss function computed on the noisy distribution in order to learn a noise robust classifier for the clean target domain.

Such a classifier can be learnt solving the following minimisation problem:

$$\arg\min R_{\beta,D_\rho}(f) = \arg\min_{f\in\mathcal{F}} \frac{1}{n}\sum_{i=1}^{n}\beta(X_i,\hat{Y}_i)\ell(f(X_i),\hat{Y}_i) \tag{32}$$

In other terms the empirical risk minimisation problem is reformulated so that the loss function takes into account how likely it is for a label to be flipped in D_ρ. Indeed, as previously shown, β represent the ratio between the clean and noisy posterior. Hence, ultimately, the more a label is affected by noise, the higher the weight assigned to it.

Figure 1 clarifies how the loss reweighting method is implemented when training a Deep Neural Network for such a classification task.

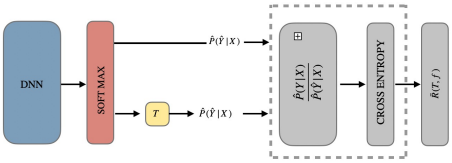

Figure 1: Loss reweighting implementation diagram.

12.2 Forward Loss Correction

An alternative approach is forward loss correction. Indeed, if we were to simply train a model on the dataset affected by class-conditional noise, we would build a classifier that predicts the flipped labels. To stir the learning process we can instead correct the prediction of our model with T before computing the loss $[P(\tilde{Y} = 1|X), ..., P(\tilde{Y} = C|X)]^T = T[P(Y = 1|X), ..., P(Y = C|X)]^T$. In this case the loss function computes the distance between the noisy prediction adjusted by T with the noisy labels as shown in Figure 2.

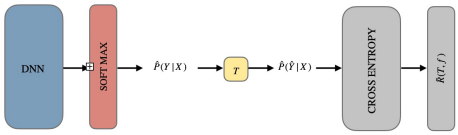

Figure 2: Forward correction implementation diagram.

In order to define the loss function computed via forward correction we now introduce an invertible link function:

$$\psi : \Delta^{c-1} \to \mathbb{R}^c$$

Such a link function can be used to define a family of loss function called composite:

$$\ell_\psi(y, h(x)) = \ell(y, \psi^{-1}(h(x)))$$

Leveraging the two previous definition we can define the forward loss correction as follows [2]:

$$\ell_\psi(h(x)) = \ell(T^\top \psi^{-1}(h(x))) \tag{33}$$

For which it can be proven that the minimiser is the same as that of the original loss function derived from the clean distribution [2].

12.3 Backward Loss Correction

The final technique that we implement in this report is backward loss correction. Intuitively, this method exploits the "memory-less" property of a Markov process [2]. In such a process, each and every state solely depends on the immediately preceding state. Similarly, we can imagine a noise process that goes from an initial state to a successive one by probabilistically flipping the value of the labels. Figure 3 illustrates this setting.

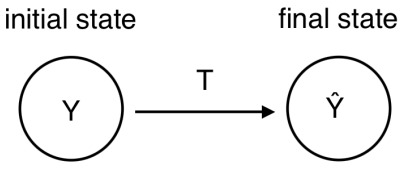

Figure 3: Markov Chain Process.

The information regarding the probability of a label being flipped is encompassed by the transition matrix T. Under the assumption that the noisy label is independent with the instance, the following expression holds:

$$\underbrace{\begin{pmatrix} P(\tilde{Y}=1|x) \\ \vdots \\ P(\tilde{Y}=C|x) \end{pmatrix}}_{\text{Noisy Class Posterior}} = \underbrace{\begin{pmatrix} P(\tilde{Y}=1|Y=1,x) & \cdots & P(\tilde{Y}=1|Y=C,x) \\ \vdots & \ddots & \vdots \\ P(\tilde{Y}=C|Y=1,x) & \cdots & P(\tilde{Y}=C|Y=C,x) \end{pmatrix}}_{T} \underbrace{\begin{pmatrix} P(Y=1|x) \\ \vdots \\ P(Y=C|x) \end{pmatrix}}_{\text{Clean Class Posterior}}$$

In order to retrieve the clean class posterior (not corrupted by noise) we need to reverse the process by applying the inverse of the transitional matrix $[P(Y=1|X),...,P(Y=C|X)]^T = T^{-1}[P(\tilde{Y}=1|X),...,P(\tilde{Y}=C|X)]^T$. The practical implementation of the aforementioned process is clarified in Figure 4. Note that while T is almost surely invertible, it can be multiplied by the identity matrix in advance to guarantee it is invertible [2].

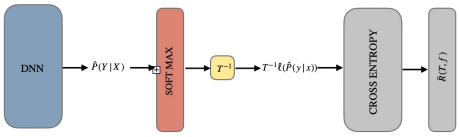

Figure 4: Backward correction implementation diagram.

Hence, we obtain the following adjusted loss function,

$$\mathcal{L}^{\leftarrow}(\tilde{y}|x) = T^{-1}\mathcal{L}(\tilde{y}|x) \tag{34}$$

For which it can be proved that the minimiser is the same as that of the original loss function for the clean dataset.

Following this approach a generic loss function $\mathcal{L}$ is computed on the noisy class posterior and then adjusted by multiplying it with the inverse of the transition matrix. The adjusted loss can be shown to be unbiased and differentiable which entails that it can be easily minimised leveraging back-propagation [2].

12.4 Transition Matrix Estimation

The transition matrix is a crucial component when building a noise robust classifiers as it can be leveraged to correct the loss function, as previously discussed. As pointed out in Section 2.2, the clean class posterior can be inferred from the noisy class posterior and transitional matrix adopting the following formula:

$$p(\tilde{Y}|X) = T^T p(Y|X) \tag{35}$$

In the previous expression the noisy class posterior can always be learnt from the noisy training. However, in many real world scenarios, T is not known apriori and hence has to be estimated.

In such a setting, where both T and $p(Y|X)$ are unknown, then they are both partially identifiable as there could be "multiple T and $p(Y|X)$ whose product equals the noisy class posterior." [16].

Hence, we need to rely on the assumption of the so-called "anchor points". An anchor point is defined as an instance X for class i if $p(Y = i|X = i)$ is equal to or close to one [9]. This estimation method also assumes the noise to be instance independent for reasons that are better explained in the following.

The notion of anchor points assumes that in the clean data domain there exist observations for which the class is almost certain. The latter allows us to ultimately find T as follows:

$$p(\tilde{Y}|X = x) = T^T p(Y|X = x) = T_i \tag{36}$$

Where the noisy class posterior can be inferred from the noisy data and can then be used to compute the anchor points (assuming they do exist).

For this equation to hold however, the mere existence of the anchor points is not enough as it implicitly assumes that T is independent of the instance (hence the noise is instance independent).

Ultimately anchor points, and thereafter the T matrix, can be learned with the following estimator:

$$x^i = \arg\max_x P(\tilde{Y} = i|x) \tag{37}$$

Such an estimator has been proven ([16]) to successfully learn the anchor points in a binary classification setup, however, it only has empirical proofs of good performances when it comes to multi-label classification tasks.

13 Experimental Setup

We investigate the effectiveness of noise-robust learning algorithms through two neural network architectures, ResNet and GoogLeNet, and over three datasets: Fashion MNIST 0.5, Fashion MNIST 0.6, and CIFAR. Our study implements forward and backward loss-correction techniques, employing transition matrices to train unbiased estimators for the clean data distribution from noisy datasets. Additionally, loss reweighting is also employed as a third method. All approaches are compared with a standard cross-entropy loss as baseline. Transition matrices are provided for MNIST 0.5 and MNIST 0.6, however we estimate the transition matrix with the anchor points method for CIFAR due to the absence of a predefined matrix.

13.1 Data Preparation and Augmentation

Our preprocessing pipeline was tailored to accommodate both RGB and greyscale images, ensuring compatibility with neural network (NN) inputs. For RGB images, we employed a series of transformations including random horizontal flips, rotations, color jitter, and normalisation to mitigate overfitting and improve model generalisability. Greyscale images underwent a similar augmentation process, with normalization adapted to single-channel data. Pre-trained models received RGB images, resized and cropped to meet the input specifications of ResNet and GoogLeNet architectures.

13.2 Model Configuration

Our experimental setup included various configurations of ResNet and GoogLeNet models, accommodating both RGB and greyscale inputs. For pre-trained models, we leveraged weights from ImageNet to expedite convergence and potentially boost performance. Each model's final layer was adjusted to reflect the number of classes in our dataset.

13.2.1 Weight Initialisation

Our model weight initialisation strategy is informed by the seminal work of [17], which addresses the challenges of training deep feed-forward neural networks. Their research identified the limitations of standard gradient descent from random initialisation, particularly the saturation of the top hidden layer when using logistic sigmoid activations. This saturation impedes learning by preventing the effective propagation of gradients through the network.

To combat these issues, Glorot and Bengio [17] proposed a novel initialisation scheme that normalises the initialisation of weights. This approach aims to maintain the variance of activations and back-propagated gradients across layers during the initial stages of training. By doing so, the network avoids the saturation of activation functions and ensures a smoother flow of gradients, which is crucial for the convergence of deeper architectures.

The proposed initialisation method, often referred to as "Xavier initialisation", has become a standard technique for setting the initial weights in neural networks, particularly when using activation functions that do not have the property of symmetry around zero, like the logistic sigmoid function. This method has been empirically shown to result in substantially faster convergence, which is why it has been adopted in our model's weight initialisation process.

13.2.2 Model Optimiser

The training phase involved iteration over the training data with a batch size of 64, where the model parameters were updated using stochastic gradient descent (momentum of 0.9 and weight decay of 0.0005). We employed a learning rate scheduler to dynamically adjust the learning rate when validation loss plateaued (reduction factor of 0.5), facilitating faster convergence. Early stopping was also implemented based on non-improving validation loss, and so while all models were given 100 epochs to train, most reached convergence earlier.

While these model decisions seemed to yield effective training, our final hyperparameters were largely unaltered from the original papers for each model, and a more extensive tuning process should be considered for future research.

14 Results

Each experiment was repeated for 10 runs, and so every estimated result below is reported as the mean over 10 runs (the result's standard deviation is also shown).

14.1 Transition Matrix Estimation Results

We present below our estimated transition matrix results for both Fashion datasets using the anchor point method. As the anchor point method requires a pre-trained model to

find the anchor points, we first trained ResNet18 and GoogLeNet on both datasets (with no noise-robustness adjustments). Following this, the model architecture that yielded the highest test accuracy was chosen to perform the estimation of the transition matrix for that dataset.

Fashion MNIST 0.5:

$$\mathbf{T} = \begin{bmatrix} 0.5 & 0.2 & 0.3 \\ 0.3 & 0.5 & 0.2 \\ 0.2 & 0.3 & 0.5 \end{bmatrix} \quad \hat{\mathbf{T}} = \begin{bmatrix} 0.519 \pm 0.024 & 0.196 \pm 0.006 & 0.285 \pm 0.021 \\ 0.303 \pm 0.022 & 0.495 \pm 0.012 & 0.201 \pm 0.017 \\ 0.224 \pm 0.023 & 0.298 \pm 0.005 & 0.478 \pm 0.022 \end{bmatrix}$$

$$\mathbf{T}_{\text{Error}} = \begin{bmatrix} 0.019 & 0.004 & 0.015 \\ 0.003 & 0.005 & 0.001 \\ 0.024 & 0.002 & 0.022 \end{bmatrix}$$

Fashion MNIST 0.6:

$$\mathbf{T} = \begin{bmatrix} 0.4 & 0.3 & 0.3 \\ 0.3 & 0.4 & 0.3 \\ 0.3 & 0.3 & 0.4 \end{bmatrix} \quad \hat{\mathbf{T}} = \begin{bmatrix} 0.403 \pm 0.021 & 0.294 \pm 0.023 & 0.303 \pm 0.005 \\ 0.286 \pm 0.024 & 0.399 \pm 0.026 & 0.315 \pm 0.009 \\ 0.298 \pm 0.020 & 0.293 \pm 0.022 & 0.410 \pm 0.003 \end{bmatrix}$$

$$\mathbf{T}_{\text{Error}} = \begin{bmatrix} 0.003 & 0.006 & 0.003 \\ 0.014 & 0.001 & 0.015 \\ 0.002 & 0.007 & 0.010 \end{bmatrix}$$

As is evident by the estimation error for both fashion transition matrices, our implemented estimation method seems to be working well, as it produces flip rate estimates very close to the known, actual, values. Considering this, we can continue to the estimation of the transition matrix for the CIFAR dataset with confidence.

CIFAR:

$$\hat{\mathbf{T}} = \begin{bmatrix} 0.789 \pm 0.084 & 0.1000 \pm 0.025 & 0.111 \pm 0.021 \\ 0.112 \pm 0.083 & 0.808 \pm 0.021 & 0.080 \pm 0.028 \\ 0.089 \pm 0.051 & 0.128 \pm 0.036 & 0.783 \pm 0.032 \end{bmatrix}$$

14.2 Classification Results

The results for each experiment are displayed below (grouped by dataset).

Fashion MNIST 0.5:

		Accuracy	**Precision**	**Recall**	**F1**
ResNet18	No T	0.923 ± 0.009	0.925 ± 0.008	0.923 ± 0.009	0.923 ± 0.009
	Forward	0.930 ± 0.002	0.932 ± 0.001	0.930 ± 0.002	0.930 ± 0.003
	Backward	0.890 ± 0.015	0.897 ± 0.011	0.890 ± 0.015	0.891 ± 0.014
	Importance Reweighting	0.925 ± 0.013	0.927 ± 0.010	0.922 ± 0.008	0.923 ± 0.012
GoogLeNet	No T	0.927 ± 0.008	0.929 ± 0.007	0.925 ± 0.008	0.927 ± 0.007
	Forward	0.937 ± 0.003	0.939 ± 0.003	0.937 ± 0.003	0.938 ± 0.003
	Backward	0.870 ± 0.025	0.876 ± 0.021	0.870 ± 0.025	0.871 ± 0.025
	Importance Reweighting	0.930 ± 0.014	0.932 ± 0.018	0.929 ± 0.021	0.928 ± 0.016

Table 1: Results for FashionMNIST0.5

Fashion MNIST 0.6:

		Accuracy	**Precision**	**Recall**	**F1**
ResNet18	No T	0.818 ± 0.022	0.822 ± 0.019	0.818 ± 0.022	0.818 ± 0.022
	Forward	0.827 ± 0.036	0.836 ± 0.028	0.827 ± 0.036	0.825 ± 0.037
	Backward	0.776 ± 0.047	0.827 ± 0.033	0.776 ± 0.047	0.778 ± 0.046
	Importance Reweighting	0.816 ± 0.004	0.818 ± 0.007	0.821 ± 0.011	0.817 ± 0.006
GoogLeNet	No T	0.771 ± 0.008	0.776 ± 0.008	0.771 ± 0.008	0.771 ± 0.008
	Forward	0.830 ± 0.044	0.836 ± 0.036	0.830 ± 0.044	0.831 ± 0.053
	Backward	0.814 ± 0.058	0.824 ± 0.085	0.814 ± 0.058	0.816 ± 0.102
	Importance Reweighting	0.816 ± 0.030	0.810 ± 0.039	0.811 ± 0.029	0.820 ± 0.011

Table 2: Results for FashionMNIST0.6

CIFAR:

		Accuracy	Precision	Recall	F1
ResNet18	No T	0.547 ± 0.025	0.549 ± 0.028	0.547 ± 0.025	0.544 ± 0.025
	Forward	0.581 ± 0.043	0.565 ± 0.045	0.581 ± 0.043	0.569 ± 0.044
	Backward	0.531 ± 0.093	0.543 ± 0.107	0.531 ± 0.093	0.516 ± 0.101
	Importance Reweighting	0.577 ± 0.036	0.582 ± 0.023	0.577 ± 0.036	0.548 ± 0.035
ResNet18 (PT)	No T	0.434 ± 0.040	0.442 ± 0.027	0.434 ± 0.040	0.420 ± 0.037
	Forward	0.487 ± 0.053	0.492 ± 0.026	0.487 ± 0.023	0.466 ± 0.041
	Backward	0.431 ± 0.093	0.423 ± 0.107	0.431 ± 0.093	0.416 ± 0.101
	Importance Reweighting	0.512 ± 0.072	0.508 ± 0.034	0.512 ± 0.072	0.502 ± 0.042
GoogLeNet	No T	0.602 ± 0.041	0.610 ± 0.046	0.602 ± 0.041	0.595 ± 0.045
	Forward	0.674 ± 0.030	0.682 ± 0.030	0.674 ± 0.030	0.671 ± 0.032
	Backward	0.594 ± 0.067	0.614 ± 0.058	0.594 ± 0.067	0.586 ± 0.075
	Importance Reweighting	0.652 ± 0.054	0.673 ± 0.026	0.652 ± 0.054	0.668 ± 0.038
GoogLeNet (PT)	No T	0.451 ± 0.042	0.453 ± 0.039	0.451 ± 0.042	0.447 ± 0.042
	Forward	0.506 ± 0.039	0.514 ± 0.023	0.506 ± 0.039	0.500 ± 0.037
	Backward	0.466 ± 0.030	0.472 ± 0.032	0.466 ± 0.030	0.458 ± 0.035
	Importance Reweighting	0.482 ± 0.028	0.496 ± 0.027	0.482 ± 0.028	0.488 ± 0.031

Table 3: Results for CIFAR

Our experimental outcomes demonstrate a notable improvement in classifier performance with the implementation of both importance reweighting, forward and backward loss correction methods. Particularly, ResNet18 and GoogLeNet exhibited enhanced accuracy and F1 scores when trained with forward correction as compared to the baseline with no transition matrix adjustment. This implies that aligning the predictions with noisy labels during training can significantly mitigate the impact of label noise.

Notably, our experimental results indicate that the backward correction method generally underperforms in comparison to both the forward correction method, importance reweighting and the baseline model without transition matrix adjustments. This observation aligns with existing research that highlights the challenges associated with backward correction in noisy label scenarios, particularly when ReLU activation is used, which can cause the loss curvature to be less sensitive to label noise [2].

Comparative analysis of the two architectures revealed that GoogLeNet marginally outperformed ResNet18 on the Fashion MNIST datasets, suggesting that the inception modules may better capture noise-robust features. However, the gap between the performances narrowed on the CIFAR dataset, indicating that ResNet18's residual learning may be more effective for complex image structures, even under noise.

15 Conclusions

This research presents a comprehensive comparison of forward correction, backward correction, and importance reweighting for handling noisy labels in classification tasks. While forward correction emerged as the leading method in terms of accuracy and F1 score, the addition of importance reweighting showcased its potential in improving

model robustness without altering the convexity of loss functions. Our methodological approach centered on the novel use of anchor points to estimate transition matrices, proving effective with minimal error margins for the Fashion MNIST datasets, which has instilled confidence in our CIFAR dataset estimations.

16 Limitation and Future Work

Our research, while pioneering in applying anchor point estimation for transition matrices in noise-robust classifiers, is not without limitations. The challenge in our approach lies in the assumption that label noise is class-conditional and independent of the data points, an assumption that may not hold in all real-world scenarios, thus potentially affecting the generalizability of our results. Additionally, the estimation of noise transition matrices becomes increasingly complex with a large number of classes, which might limit the scalability of our method.

Future work should explore the integration of importance reweighting with meta-learning frameworks to optimize label noise correction further. Additionally, assessing the performance of these methods on larger datasets with real-world noise distributions will be crucial [18].

Furthermore, we aim to explore the effectiveness of Large Language Models (LLMs) in noise-robust classification tasks, leveraging their advanced pattern recognition and generative capabilities [19].

T-revision, a method that refines the transition matrix, stands out as a promising direction for further enhancing the robustness of classifiers against label noise.

We also plan to investigate the imposition of lower bounds on the empirical risk during training, as a direct measure to mitigate overfitting induced by label noise [20].

References

[1] Hwanjun Song, Minseok Kim, Dongmin Park, Yooju Shin, and Jae-Gil Lee. Learning from Noisy Labels with Deep Neural Networks: A Survey, March 2022. arXiv:2007.08199 [cs, stat].

[2] Giorgio Patrini, Alessandro Rozza, Aditya Krishna Menon, Richard Nock, and Lizhen Qu. Making Deep Neural Networks Robust to Label Noise: A Loss Correction Approach. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2233–2241, Honolulu, HI, July 2017. IEEE.

[3] Benoit Frenay and Michel Verleysen. Classification in the Presence of Label Noise: A Survey. *IEEE Transactions on Neural Networks and Learning Systems*, 25(5):845–869, May 2014.

[4] Xingquan Zhu and Xindong Wu. Class Noise vs. Attribute Noise: A Quantitative Study. *Artificial Intelligence Review*, 22(3):177–210, November 2004.

[5] José A. Sáez, Mikel Galar, Julián Luengo, and Francisco Herrera. Analyzing the presence of noise in multi-class problems: alleviating its influence with the One-vs-One decomposition. *Knowledge and Information Systems*, 38(1):179–206, January 2014.

[6] Xuefeng Li, Tongliang Liu, Bo Han, Gang Niu, and Masashi Sugiyama. Provably End-to-end Label-Noise Learning without Anchor Points, October 2021. arXiv:2102.02400 [cs].

[7] Nagarajan Natarajan, Inderjit Dhillon, Pradeep Ravikumar, and Ambuj Tewari. Learning with Noisy Labels. *Advances in Neural Information Processing Systems (NIPS 2013)*, 26, 2013.

[8] Görkem Algan and Ilkay Ulusoy. Image Classification with Deep Learning in the Presence of Noisy Labels: A Survey. *Knowledge-Based Systems*, 215:106771, March 2021. arXiv:1912.05170 [cs, stat].

[9] Xiaobo Xia, Tongliang Liu, Nannan Wang, Bo Han, Chen Gong, Gang Niu, and Masashi Sugiyama. Are Anchor Points Really Indispensable in Label-Noise Learning?, December 2019. arXiv:1906.00189 [cs, stat].

[10] Ruxin Wang, Tongliang Liu, and Dacheng Tao. Multiclass Learning With Partially Corrupted Labels. *IEEE Transactions on Neural Networks and Learning Systems*, 29(6):2568–2580, June 2018.

[11] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks, July 2017. arXiv:1703.03400 [cs].

[12] Zhaowei Zhu, Yiwen Song, and Yang Liu. Clusterability as an Alternative to Anchor Points When Learning with Noisy Labels, July 2021. arXiv:2102.05291 [cs, stat].

[13] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, Boston, MA, USA, June 2015. IEEE.

[14] Simone Ricci, Tiberio Uricchio, and Alberto Del Bimbo. Smoothing and Transition Matrices Estimation to Learn with Noisy Labels. In Gian Luca Foresti, Andrea Fusiello, and Edwin Hancock, editors, *Image Analysis and Processing – ICIAP 2023*, volume 14233, pages 450–462. Springer Nature Switzerland, Cham, 2023. Series Title: Lecture Notes in Computer Science.

[15] Tongliang Liu and Dacheng Tao. Classification with Noisy Labels by Importance Reweighting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(3):447–461, March 2016. arXiv:1411.7718 [cs, stat].

[16] Yivan Zhang, Gang Niu, and Masashi Sugiyama. Learning Noise Transition Matrix from Only Noisy Labels via Total Variation Regularization, June 2021. arXiv:2102.02414 [cs, stat].

[17] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. *Journal of Machine Learning Research*, 9:249–256, January 2010.

[18] Y X and W Z. Classification with noisy labels by importance reweighting. *arXiv preprint arXiv:1411.7718*, 2015.

[19] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness, Jun 2022.

[20] William Toner and Amos Storkey. Label noise: Correcting a correction, 2023. [Online; accessed 10-November-2023].

Appendix A: Code Running Instructions

1. Place .npz data files in /code/data
2. Run whole notebook "COMP5328_A2_Code.ipynb".
3. To alter experiment, all that must be changed is arguments of run_pipeline().