Assignment 2

Professor: Tongliang Liu
Group members: Alex Elias (UniKey: aeli0392, SID: 500407020) & Luca Quaglia (UniKey: lqua3126, SID: 500175059)

Abstract

Modern classification algorithms are designed to estimate labels based on data. Problems arise if the labels are untrustworthy or misleading. This paper takes a probabilistic approach to treating mislabelled given some assumptions about the distribution of mislabelled data. We solve two problems in such a manner: estimating the distribution of noisy data (given the assumptions) and training models given noisy data.

1 Introduction

Modern machine learning applications are fed datasets that have millions of labelled examples, however accurately labelling millions of images is expensive, often to the point of implausibility. Problems arise in areas such as object recognition - where the vast amount of data needed makes the problem expensive [6] or medical imaging where experts are needed and can be unreliable increasing the cost of individual labels.

Using the simple assumption that the probability of a label being incorrect is a function of only the true class, we call this assumption the assumption of independance of X from noise and is denoted by

$$P\left(\text{noise}|X,Y\right) = P\left(\text{noise}|Y\right) \tag{1}$$

we will illustrate a method of estimating these probabilities for each class and how they can be used to improve model performance on data that is extremely noisy (with up to half of the labels being incorrect).

We will then introduce a method to calculate an estimate $\widehat{P}\left(\widetilde{Y}\middle|Y\right)$ using a real-world dataset - a subset of 3 classes from the fashion MNIST dataset[1] with introduced synthetic noise (that follows the assumption outlined above), and we will evaluate the effectiveness of this method by comparing with the true noise distribution that we generated the labels with. This entire process will be repeated for two different noise distributions to ensure the generalisation of the algorithm.

We will then use our method on noise distribution estimation to estimate the noise distribution on a subset of a second dataset - the CIFAR dataset [2].

Given noise distributions, we will then demonstrate how they can be used to train models that will perform very well considering the noise rates (often predicting the correct label more frequently then the noisy labels provided).

1.1 Notation in this report

Throughout this report we will denote $(\mathcal{X}, \mathcal{Y})$ as the domain of datasets (X, Y) for which each observation $(X_i, Y_i) \sim P(X, Y)$ Labels with introduced noise will be denoted $\widetilde{Y}$ while clean labels will be denoted Y. This implies that $P\left(Y_i = \widetilde{Y}_i\right) = P(\text{noise}|Y)$ if the assumption of independance of X from noise holds. Any time we have a probability or distribution that is an estimate we will denote it with $\widehat{P}(\cdot)$

Once models enter the arena, we will denote their predictions as $\widehat{Y}$. The *true* transition matrix is denoted as T while its estimate is denoted by $\widehat{T}$. The true value of T is given by:

$$T = \begin{bmatrix} P\left(\widetilde{Y}=1\middle|Y=1\right) & P\left(\widetilde{Y}=1\middle|Y=2\right) & \dots & P\left(\widetilde{Y}=1\middle|Y=d\right) \\ P\left(\widetilde{Y}=2\middle|Y=1\right) & P\left(\widetilde{Y}=2\middle|Y=2\right) & \dots & P\left(\widetilde{Y}=2\middle|Y=d\right) \\ \vdots & \vdots & \ddots & \vdots \\ P\left(\widetilde{Y}=d\middle|Y=1\right) & P\left(\widetilde{Y}=d\middle|Y=2\right) & \dots & P\left(\widetilde{Y}=d\middle|Y=d\right) \end{bmatrix} \tag{2}$$

and is occasionally referred to as the *distribution of noise on* X while $\widehat{T}$ is given by

$$\widehat{T} = \begin{bmatrix} \widehat{P}\left(\widetilde{Y}=1\middle|Y=1\right) & \widehat{P}\left(\widetilde{Y}=1\middle|Y=2\right) & \dots & \widehat{P}\left(\widetilde{Y}=1\middle|Y=d\right) \\ \widehat{P}\left(\widetilde{Y}=2\middle|Y=1\right) & \widehat{P}\left(\widetilde{Y}=2\middle|Y=2\right) & \dots & \widehat{P}\left(\widetilde{Y}=2\middle|Y=d\right) \\ \vdots & \vdots & \ddots & \vdots \\ \widehat{P}\left(\widetilde{Y}=d\middle|Y=1\right) & \widehat{P}\left(\widetilde{Y}=d\middle|Y=2\right) & \dots & \widehat{P}\left(\widetilde{Y}=d\middle|Y=d\right) \end{bmatrix} \tag{3}$$

and denotes the *estimated noise distribution on* X.

d will denote the dimensionality of the data such that $X_i \in \mathbb{R}^d$.

δ_{ab} will denote the Kronecker delta function defined as

$$\delta_{ab} = \left\{ \begin{array}{ll} 1 & : a = b \\ 0 & : a \neq b \end{array} \right|$$

The domain of probability vectors of length k will be denoted $\mathcal{P}^k$, where $p \in \mathcal{P}^k : \sum p = 1, p_i \geq 0$. It is worth noting that any δ_{ij} meets these criteria.

2 Related Work

There have been many studies into perceptron designs that can tolerate noise following the assumption of independance of X from noise . A survey of noise-tolerant variants of perceptron algorithms found varying amounts of success limited to the perceptron estimator domain. Learning from datasets affected by label noise has been studied in various ways and in different domains. Noise of three types has been investigated: random classification noise, class dependent noise and class and instance dependent noise. The first type of noise is the most studied and analysed. Very early on, at the end of the '80s, it was shown that random classification noise increases the number of examples for identifying a model in the probably approximately correct framework [5]. To deal with random classification noise one avenue is to develop robust models and robust surrogate loss functions have been developed. Another avenue is to try to filter out noisy examples. This method has the drawback of potentially eliminating many useful examples. Still another avenue is to explicitly model the effect of label noise and inferring its distribution [7]. The second type of noise has been studied more recently [8] and it is the framework used in this work. Liu and Tao proposed a way to exploit the statistical characteristics of the label noise to re-weight the probabilities for the various classes outputted by a classification model. The re-weighting of the probabilities can be achieved by a forward or a backward method [4]. Given an estimate of the transition matrix transforming probabilities for clean labels into probabilities for noisy labels, the forward method inserts the transition

matrix between the output of the model and the loss function. The net effect is to train a model that is suitable for classifying clean labels, even if the loss function is computed on the noisy data-set. In the backward method the inverse of the transition matrix is used to get the clean labels probabilities after training the model on the noisy data-set.

3 Methods

3.1 Problem Setup

We have taken 2 datasets - a subset of 3 classes from the fashion MNIST dataset [1] and a subset of 3 classes from the CIFAR dataset [2]. Each dataset was taken with two subsets - a subset with synthetic noise (that followed some distribution $P\left(\widehat{Y}\middle|Y\right)$) and a subset of the data that had entirely clean labels.

The Fashion MNIST was used twice, each time with different distributions of noise, where the noise is known. The training subset (subset with synthetic label noise) contained 18000 training examples of 28×28 and the clean subset (subset with no label noise) had 3000 examples.

The CIFAR dataset was only used once, with an unknown noise distribution. The noisy subset had 15000 training examples (subset with synthetic label noise) and the testing subset (without noise) had 3000.

3.2 Models used

In this analysis, we use two models to demonstrate the general applications of this method of noise distribution estimation and model training given a noise distribution. Models that train with an iterative process are clearer to demonstrate mathematically so we have decided to use two neural networks - a fully connected neural network and a convolutional neural network.

3.2.1 Fully Connected Neural Network

The **F**ully **c**onnected **n**eural **n**etwork (FCNN) we used used the pyramid rule[9] to decide the number of hidden nodes at each layer. The relu activation function [3] was used between layers. For the MNIST datasets we used 2 hidden layers of sizes 123 and 19. A SoftMax computed the final class probabilities.

The FCNN used for the CIFAR dataset had 4 hiden layers of sizes 305, 305, 30, 30. A SoftMax computed the final class probabilities.

3.2.2 Convolutional Neural Network (CNN)

The convolutional network structure we used had a convolutional layer (6 filters, size 5, stride 1), an average pooling layer (size 2, stride 2), a convolutional layer (10 filters, size 5, stride 1), an average pooling layer (size 2, stride 2) and two fully connected layers (size 160 and 22) followed by a SoftMax.

Any time we train a model, we will use only the training set (noisy labels) cross validated with 10 folds. This will allow us to compute point estimates and standard deviations for all values to decrease the likelihood of getting 'unlucky' - having an unlikely result represent our findings.

3.3 Distribution of noise estimation

Suppose we have access to only a noisy dataset, $\left(X, \widetilde{Y}\right)$ and we want to determine $P\left(\widetilde{Y}\middle|Y\right)$. Let us also suppose that the dataset is sufficiently large that we can estimate its noisy label distribution

conditional on X, $P\left(\widetilde{Y}\middle|X\right)$. With the application of the law of total probability:

$$P\left(\widetilde{Y}\middle|X\right) = \sum_{c \in Y} P\left(\widetilde{Y}\middle|X, Y=c\right) P\left(Y=c|X\right)$$

But, with the assumption of independance of X from noise (Equation 1) we have

$$P\left(\widetilde{Y}\middle|X\right) = \sum_{c \in Y} P\left(\widetilde{Y}\middle|Y=c\right) P\left(Y=c|X\right) \tag{4}$$

If we can find an observation x for which its class is known to be t, IE $P\left(Y=c|X=x\right) = \delta_{ct}$ then 4 simplifies to

$$P\left(\widetilde{Y}\middle|X=x\right) = P\left(\widetilde{Y}\middle|Y=t\right) \tag{5}$$

In this way, we call x an *anchor point* as it allows us to predict the probability distribution of noise on some class. In reality, we do not have access to such a datapoint *or* the true distribution of noise, so we estimate them using a model.

Theoretically, any model that can output a probability distribution can be used as $\widehat{P}\left(\widetilde{Y}\middle|X\right)$, which would result in the noise distribution estimate $\widehat{P}\left(\widetilde{Y}\middle|Y\right)$. Our neural network implementations fit the description (provided a final activation layer provides a 1-summing vector of the correct length - we used the softmax function)

If we suppose that the *expected* class for an observation is known, we can use the law of large numbers to include multiple anchor points per class and averaging the resultant probability distributions[1].

3.4 Model training with known distribution of noise

Now, suppose we have a known distribution of label noise $P\left(\widetilde{Y}\middle|Y\right)$, in the form of a transition matrix T and a model structure, $f_\theta(x) : \mathcal{X} \mapsto \mathcal{P}^d$ where $\mathcal{X}$ is the problem domain. Let us further suppose that we have a surrogate loss function $l\left(\widehat{Y}, Y\right)$ which is used to optimize the parameters θ of f to minimize the loss function.

Training models in this way normally takes the form of $\arg\min_\theta l(f_\theta(X), Y)$ to achieve the best predictor for Y. For this problem however, we only have $\widetilde{Y}$ and fitting $\arg\min_\theta l\left(f_\theta(X), \widetilde{Y}\right)$ would result in a model that attempts to predict $\widetilde{Y}$. To solve this problem, we treat $\widetilde{Y}$ as a probability distribution $P\left(\widetilde{Y}\middle|X\right)$. Plugging into equation 4, we get a way to convert $P\left(Y|X\right)$ to $P\left(\widetilde{Y}\middle|X\right)$. Notice, that this is equivalent to $T \cdot \left(P\left(Y_1|X\right), P\left(Y_2|X\right), \ldots, P\left(Y_d|X\right)\right)^T$. Using this information, we can re-weight the output of the model f in the loss function l to optimize for the non-noisy label space instead of the noisy labels:

$$l\left(T \cdot f_\theta(X), \widetilde{Y}\right) \tag{6}$$

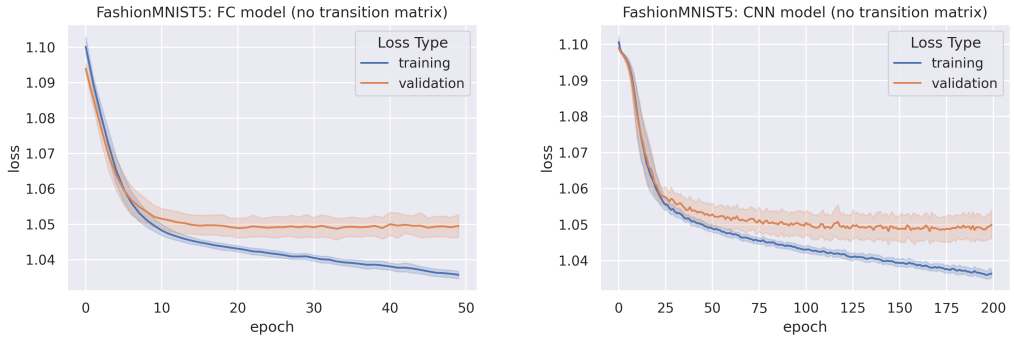
To take an intuition from the mathematics, the model attempts to predict *the correct column* of the transition matrix to return, instead of the noisy label. Since $E_{X=x}\left[\widetilde{Y}\right] = P\left(\widetilde{Y}\middle|X=x\right) =$ *column Y_i of the transition matrix*, the function is rewarded for estimating the true Y label (the index of the column)

[1]Future studies could observe the effects of the weighted averaging based on the confidence of the model, or on the number of standard deviations that anchor points distribution is from the mean of the estimate.

4 Experiments

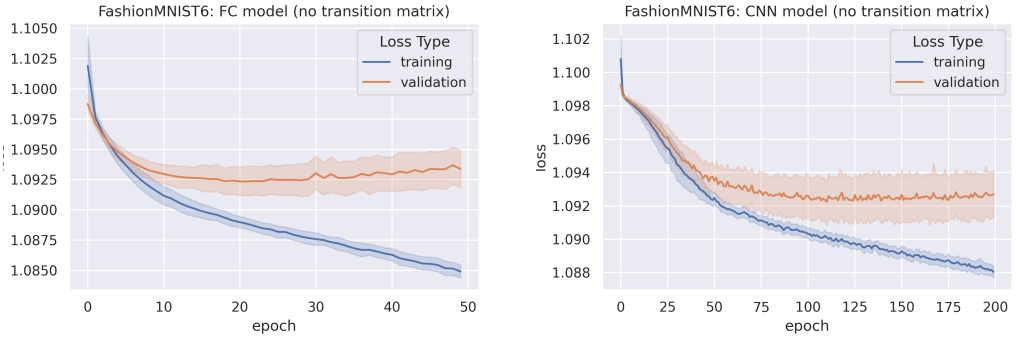
4.1 Estimating the distribution of noise

We trained the FCNN and CNN models on the noisy dataset for both MNIST5 and MNIST6. The training and validation loss for both models are presented in figures 1 and 2. As expected, the models were unable to obtain a high accuracy before beginning to overtrain[2]. The accuracy plot for the FC model (figures 10 and 11) show a similar story, but much less pronounced[3]. This gives us an idea of when the model is no longer representing the distribution of the data and is training on noise within the specific examples in the training set, meaning the difference (by any metric other then 0-1 loss) is large.



(a) Fully Connected Neural Network (b) Convolutional Neural Network

Figure 1: Epoch under loss of both models on the FashionMNIST5 dataset, trained without transition matrix FashionMNIST5no$_T loss$



(a) Fully Connected Neural Network (b) Convolutional Neural Network

Figure 2: Epoch under loss of both models on the FashionMNIST6 dataset, trained without transition matrix FashionMNIST6no$_T loss$

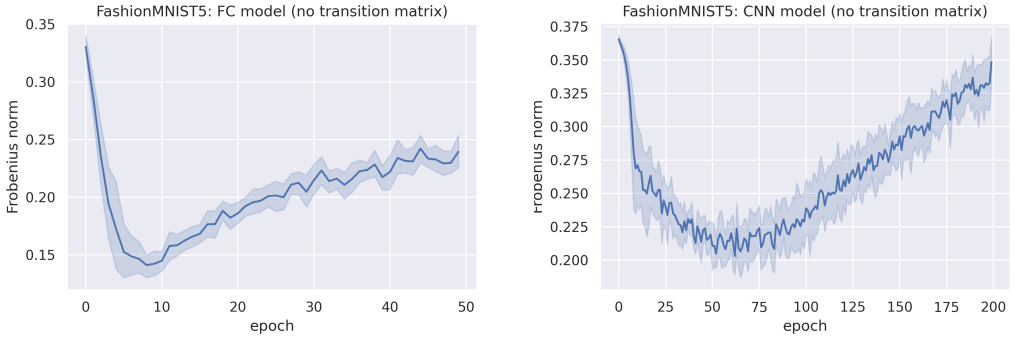
We used the estimation procedure defined in section 3.3 to estimate the noise. As intuition would dictate, observing the difference between $\widehat{T}$ and $T = \left\|\widehat{T} - T\right\|_{2,2}$ (figures 3 and 4) show that the estimates becomes closest to the true value at around the same time as the validation loss stops

[2]This is expected because 50% and 60% of the labels do not conform to the data distribution, instead being random making them not dependant on the data - the model has no hope other then to 'guess' them. Theoretically, the maximum accuracy it could obtain would be to correctly guess the 50% or 40% labels + guess the others correctly $\frac{1}{3}$ of the time

[3]This may be due to the effect of taking the argmax and ignoring the absolute values, the model is predicting probabilities that are not very high on the validation set and high probabilities on the training set.

decreasing.

The estimates achieved were fairly close to the true values as shown in figure 13 and 14.



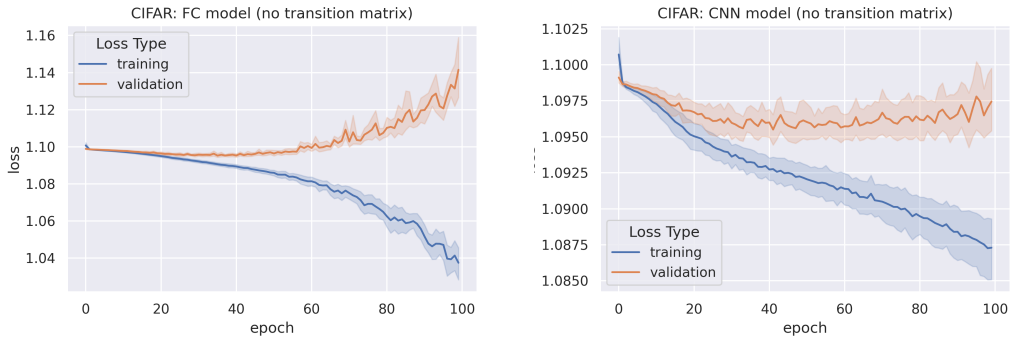
(a) Fully Connected Neural Network (b) Convolutional Neural Network

Figure 3: Epoch under frobenius norm of both models on the FashionMNIST5 dataset FashionMNIST5no$_T frobenius$

(a) Fully Connected Neural Network (b) Convolutional Neural Network

Figure 4: Epoch under frobenius norm of both models on the FashionMNIST6 dataset FashionMNIST6no$_T frobenius$

Using the knowledge gained above about how to estimate the noise distribution, we are able to do the same process without the benefit of the frobinius norm. By taking the transition matrix when the validation loss is no longer decreasing, a fairly accurate estimate of the distribution is achieved. Figure 5 shows the loss during training on the noisy labels for the CIFAR dataset. Estimates of the transition matrix were taken at 30 epochs for both models, as this is where the models stopped improving. Figure 15 shows the estimates are quite similar. Figure 12 shows the training and validation accuracy over epochs.



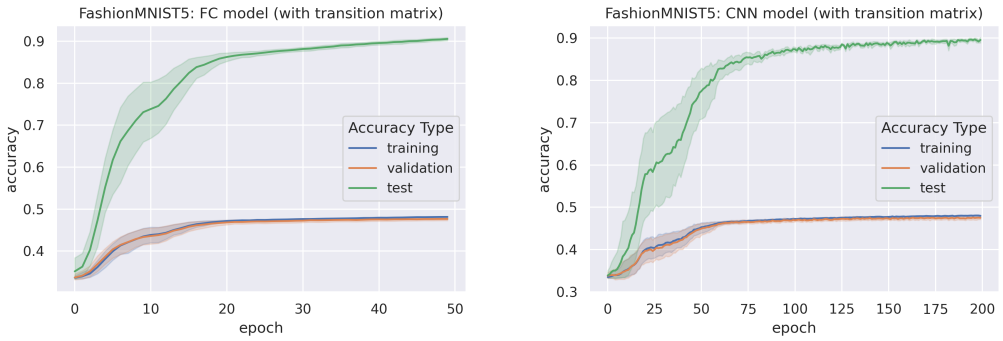
(a) Fully Connected Neural Network

(b) Convolutional Neural Network

Figure 5: Epoch under loss of both models on the CIFAR dataset, trained without transition matrix CIFARno$_T loss$

4.2 Model performance with known distribution of noise

Once the noise distributions are known, we can modify the loss function as defined in equation 6 to train the model to predict the true labels. Figures 6 and 7 show the accuracy on the test set over epochs. The results are astonishing, getting about 90% of the predictions correct for both model architectures. It appears as though the FashionMNIST6 CNN model has one or two models that have not finished training (figure 7b) but it is still up at around 60% and still increasing. Figures 20 and 21 show the training loss, which tells a similar story.



(a) Fully Connected Neural Network (b) Convolutional Neural Network

Figure 6: Epoch under accuracy of both models on the FashionMNIST5 dataset, trained with transition matrix FashionMNIST5with$_T accuracy$

(a) Fully Connected Neural Network (b) Convolutional Neural Network

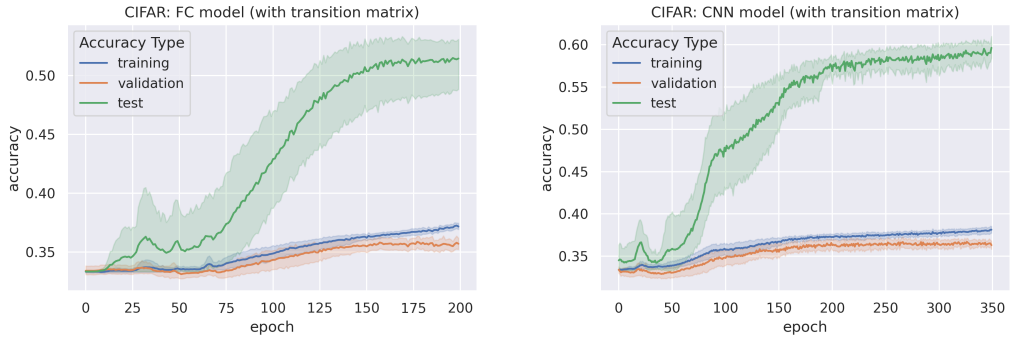
Figure 7: Epoch under accuracy of both models on the FashionMNIST6 dataset, trained with transition matrix FashionMNIST6with$_T accuracy$

4.3 Model performance with estimated distribution of noise

The models were trained to predict the noisy CIFAR examples as well. With this dataset however we did not have access to the true distribution, so we must use our estimate. Each model used the estimate it estimated. The results are shown in 8 and 9 for accuracy and loss respectively.

We also ran the FashionMNIST datasets with their estimated noise distributions. Figures 16 and 17 show the losses and Figures 18, 19 show the accuracy. It is clear that the estimation matrices are *good enough* to achieve much higher accuracy then if you were to not use our method.

These results makes it clear that even through no known examples of values for the dataset, the methods described in the methods section of this paper can achieve a much better then the almost random guessing you would get with the FashionMNIST6 and FashionMNIST5 datasets. The estimated noise distribution for CIFAR also indicated that the labels were quite close to



(a) Fully Connected Neural Network (b) Convolutional Neural Network

Figure 8: Epoch under accuracy of both models on the CIFAR dataset, trained with transition matrix CIFARwith$_T accuracy$

(a) Fully Connected Neural Network (b) Convolutional Neural Network

Figure 9: Epoch under loss of both models on the CIFAR dataset, trained with transition matrix CIFARwith$_T loss$

random, (the noise estimates almost always under-estimated the amount of noise, as is evident in figures 13 and 14)

5 Conclusion

We have outlined methods to train models with noisy labels given a noise distribution, and we have also provided a method to predict a noise distribution if it is unknown (as is most likely the case in real-world scenarios). We have demonstrate these methods capabilities through 3 different datasets, two where the noise is known and 1 where the noise is unknown. Even with the unknown noise (and the more difficult to predict dataset) we achieve an accuracy of 60% with low dimensionality.

Future work can be done in both methods. Most notably, the method of noise distribution estimation relies on anchor points to estimate the columns of the transition matrix. We have assumed that the anchor points that are estimated with highest probability by the model the predictions are the same as the labels, but this is not necessarily the case. If the distributions on Y are very different to other classes (as was the case for the FashionMNIST5 dataset) this may have a detremental effect on the prediction of the noise distributions. We would like to revisit this in the future to examine adversarial-like attacks on this method to see how robust it is to this problem.

We took the average of 10 highest probability classes on the anchor points to estimate the transition matrix, but did not examine the standard deviation within estimates for the transition matrix (only between cross validation folds). This may provide some further insight into the problem.

A Appendix: Code instructions

We have provided 6 Google Colab notebooks: two for each data-set, one for the fully connected neural network and one for the convolutional neural network.The code expects the datasets in a subfolder called ”dataset”.

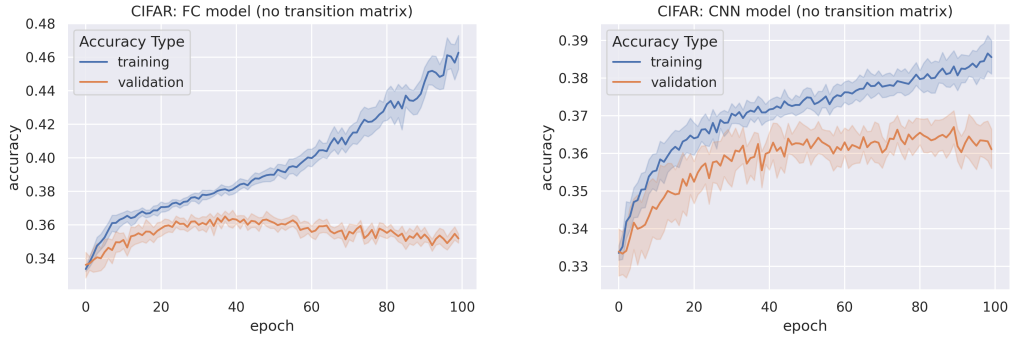
A.1 Figures

(a) Fully Connected Neural Network (b) Convolutional Neural Network

Figure 10: Epoch under accuracy of both models on the FashionMNIST5 dataset, trained without transition matrix FashionMNIST5no$_T accuracy$

(a) Fully Connected Neural Network (b) Convolutional Neural Network

Figure 11: Epoch under accuracy of both models on the FashionMNIST6 dataset, trained without transition matrix FashionMNIST6no$_T accuracy$



(a) Fully Connected Neural Network (b) Convolutional Neural Network

Figure 12: Epoch under accuracy of both models on the CIFAR dataset, trained without transition matrix CIFARno$_T accuracy$

	FCNN	CNN	Actual
Mean	$\begin{bmatrix}0.56 & 0.16 & 0.27\\ 0.27 & 0.58 & 0.20\\ 0.17 & 0.26 & 0.53\end{bmatrix}$	$\begin{bmatrix}0.60 & 0.16 & 0.25\\ 0.25 & 0.60 & 0.16\\ 0.15 & 0.24 & 0.59\end{bmatrix}$	$\begin{bmatrix}0.5 & 0.2 & 0.3\\ 0.3 & 0.5 & 0.2\\ 0.2 & 0.3 & 0.5\end{bmatrix}$
SD	$\begin{bmatrix}0.02 & 0.01 & 0.01\\ 0.03 & 0.02 & 0.02\\ 0.03 & 0.02 & 0.03\end{bmatrix}$	$\begin{bmatrix}0.03 & 0.01 & 0.03\\ 0.02 & 0.03 & 0.02\\ 0.02 & 0.02 & 0.03\end{bmatrix}$	
from Epoch	10	75	

Figure 13: Estimated FashionMNIST5 noise distribution

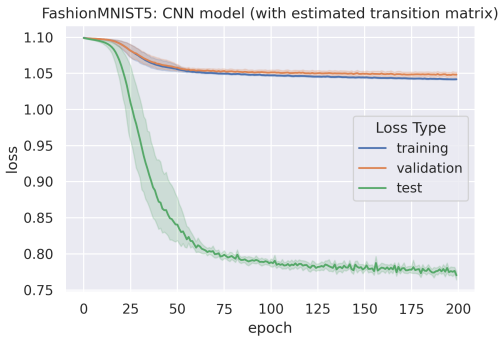
	FCNN	CNN	Actual
Mean	$\begin{bmatrix}0.41 & 0.30 & 0.30\\ 0.29 & 0.40 & 0.31\\ 0.30 & 0.30 & 0.39\end{bmatrix}$	$\begin{bmatrix}0.44 & 0.29 & 0.30\\ 0.27 & 0.44 & 0.28\\ 0.29 & 0.27 & 0.42\end{bmatrix}$	$\begin{bmatrix}0.4 & 0.3 & 0.3\\ 0.3 & 0.4 & 0.3\\ 0.3 & 0.3 & 0.4\end{bmatrix}$
SD	$\begin{bmatrix}0.03 & 0.02 & 0.02\\ 0.02 & 0.02 & 0.01\\ 0.02 & 0.02 & 0.02\end{bmatrix}$	$\begin{bmatrix}0.01 & 0.01 & 0.02\\ 0.01 & 0.01 & 0.03\\ 0.01 & 0.02 & 0.02\end{bmatrix}$	
from Epoch	8	60	

Figure 14: Estimated FashionMNIST6 noise distribution

	FCNN	CNN	Actual
Mean	$\begin{bmatrix}0.40 & 0.33 & 0.29\\ 0.34 & 0.38 & 0.29\\ 0.26 & 0.29 & 0.41\end{bmatrix}$	$\begin{bmatrix}0.41 & 0.32 & 0.27\\ 0.33 & 0.40 & 0.28\\ 0.27 & 0.28 & 0.44\end{bmatrix}$	(unknown)
SD	$\begin{bmatrix}0.02 & 0.04 & 0.02\\ 0.02 & 0.02 & 0.02\\ 0.03 & 0.03 & 0.02\end{bmatrix}$	$\begin{bmatrix}0.03 & 0.02 & 0.02\\ 0.02 & 0.03 & 0.02\\ 0.02 & 0.02 & 0.04\end{bmatrix}$	
from Epoch	30	30	

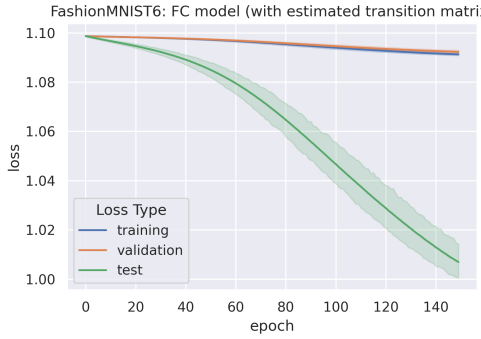
Figure 15: Estimated CIFAR noise distribution

(a) Fully Connected Neural Network

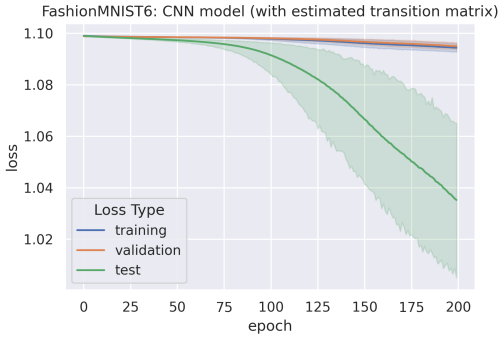


(b) Convolutional Neural Network

Figure 16: Epoch under loss of both models on the FashionMNIST5 dataset, trained with transition matrix FashionMNIST5with$_T eloss$

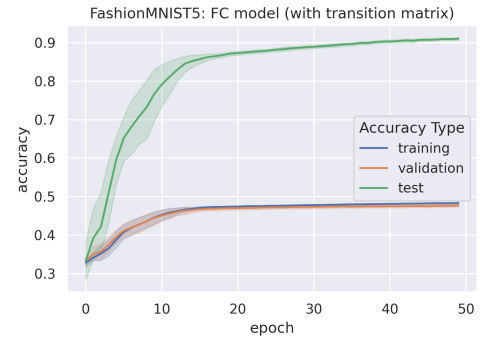


(a) Fully Connected Neural Network

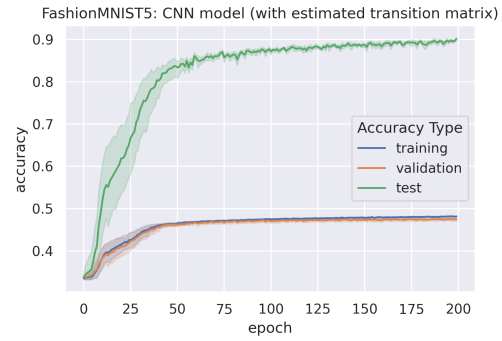


(b) Convolutional Neural Network

Figure 17: Epoch under loss of both models on the FashionMNIST6 dataset, trained with transition matrix FashionMNIST6with$_T eloss$



(a) Fully Connected Neural Network

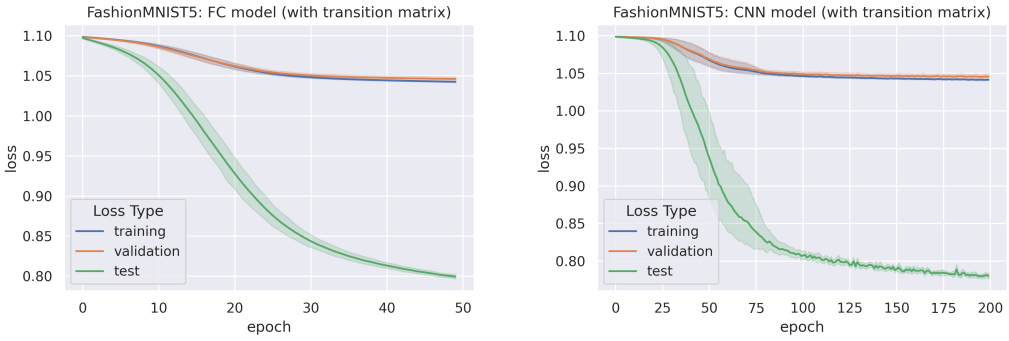


(b) Convolutional Neural Network

Figure 18: Epoch under accuracy of both models on the FashionMNIST5 dataset, trained with transition matrix FashionMNIST5with$_T eaccuracy$

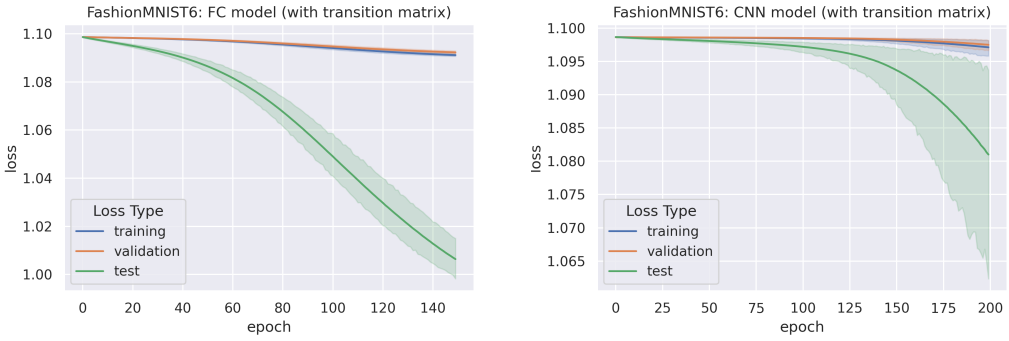
(a) Fully Connected Neural Network (b) Convolutional Neural Network

Figure 19: Epoch under accuracy of both models on the FashionMNIST6 dataset, trained with transition matrix FashionMNIST6with$_T eaccuracy$



(a) Fully Connected Neural Network (b) Convolutional Neural Network

Figure 20: Epoch under loss of both models on the FashionMNIST5 dataset, trained with transition matrix FashionMNIST5with$_T loss$



(a) Fully Connected Neural Network (b) Convolutional Neural Network

Figure 21: Epoch under loss of both models on the FashionMNIST6 dataset, trained with transition matrix FashionMNIST6with$_T loss$

References

[1] URL: https://www.kaggle.com/zalando-research/fashionmnist.

[2] URL: https://www.kaggle.com/c/cifar-10.

[3] Abien Fred Agarap. "Deep Learning using Rectified Linear Units (ReLU)". In: (2019). arXiv: 1803.08375 [cs.NE].

[4] G. Patrini et al. "Making deep neural networks robust to label noise: a loss correction approach". In: *arXiv:1609.03683* (2017).

[5] D. Angluin and P. Laird. "Learning from noisy examples". In: *Machine Learning* (1999).

[6] S. Belongie D. Rolnick A. Veit and N. Shavit. "Deep Learning is Robust to Massive Label Noise". In: *arXiv:1705.10694* (2018).

[7] B. Frenay and M. Verleysen. "Classification in the presence of label noise: a survey". In: *IEEE Transactions on Neural Networks Learning Systems* (2014).

[8] Tongliang Liu and Dacheng Tao. "Classification with Noisy Labels by Importance Reweighting". In: (Sept. 2016).

[9] T.Masters. *Practical Neural Networks in C++*. 1993.