Assignment 2

Tutors: Yu Yao, Xuefeng Li, Songhua Wu

Group members:

Zhecheng Guo	490574782	zguo6620
Junfei Gu	480503075	jugu0751
Yuan Gao	500158878	ygao2778

Abstract

It is essential to solving the label noise problem in large data sets. This report will introduce the robustness of Convolution Neural Network (CNN) and VGG16 classifiers on three data sets containing random label noise. We will use the known transition matrix for the first two data sets, and for the last data set, we will implement an estimator to estimate the transition matrix. In the Introduction, we will introduce the meaning and function of the label noise problem; Related work, we will review the algorithms used in the past and their advantages and disadvantages; In the Method section, we will share the details of the classification model. The last part is the experiment and conclusion.

1 Introduction

In machine learning, the most common problem is classification. There are already many classifiers that complete classification tasks. We put the sample data set into the classifier and obtain coefficients or weights to predict other data. However, random label noise is prevalent in various large-scale data sources (such as sensor networks and the Web). They can easily cause samples with incorrect labels in the data set and severely reduce the accuracy of the classifier. The fact that machine learning algorithms behave differently under different settings makes it more important to solve the label noise problem in training large data sets.

In this report, we will use the Convolution Neural Network (CNN) and VGG16 classifier. The benefit of CNN is to extract image features and convert them to lower dimensions without losing their parts [1]. As a particular architecture of CNN, VGG16 is considered one of the best visual model architectures so far. The uniqueness of VGG16 does not have a large number of hyperparameters. It instead focuses on the convolutional layer. By using a 3x3 filter with stride1, and always uses the same filling and maxpool layer of a 2x2 filter of stride2. The convolution and max-pooling layers are consistent throughout the architecture. This feature allows VGG16 to improve the accuracy of the training set better and reduce a loss [2].

2 Related Work

Before CNN, decision trees were one of the standard algorithms in machine learning, computer vision, information retrieval and other fields. The Random Forest has a history of success in solving machine learning tasks. It has many attractive features, such as simple structure, good ability to process high-dimensional data, and relatively good speed performance in training and testing [3]. Therefore, random forests have been used in many computer vision applications and have produced many new results.

Although the Random Forest has many superior properties, it also has certain inefficiencies. Since in the random forest, the evaluation of each split node is based on the purity of the node, the label noise in the training data may affect the purity evaluation, thereby reducing the performance [4]. There have been some efforts to improve the performance of random forests with noisy labels. In the case of symmetric label noise and a large sample size per node, the decision tree and random forest algorithms are both robust. In asymmetric noise, they are usually not robust. Unfortunately, in most practical applications, label noise may be very asymmetric, so in the face of label noise, improving the robustness of random forests is still an unsolved problem [4]. Therefore, we will use CNN and VGG16 to make up for this deficiency.

3 Method

3.1 Transition Matrix

The transition matrix is flip rate which is related with Y and $\widetilde{Y}$.Ymeans clean label and $\widetilde{Y}$means label noise label.

Under the class-dependent assumption:

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

Transition matrix can be represented as follows:

$$\begin{bmatrix} P\left(\widetilde{Y}=1\middle|Y=1\right) & P\left(\widetilde{Y}=1\middle|Y=2\right) & \cdots & P\left(\widetilde{Y}=1\middle|Y=C\right) \\ P\left(\widetilde{Y}=2\middle|Y=1\right) & P\left(\widetilde{Y}=2\middle|Y=2\right) & \cdots & P\left(\widetilde{Y}=2\middle|Y=C\right) \\ \vdots & \vdots & \vdots & \vdots \\ P\left(\widetilde{Y}=C\middle|Y=1\right) & P\left(\widetilde{Y}=C\middle|Y=2\right) & \cdots & P\left(\widetilde{Y}=C\middle|Y=C\right) \end{bmatrix} \tag{2}$$

Anchor point it a great way to estimate Transition matrix. For the i_th class, one 'x' is called an anchor point and $P\left(\widetilde{Y}=i\middle|X=x\right)=1$. The formula is:

$$P\left(\widetilde{Y}=j\middle|X=x\right) = \sum_{k=1}^{C} T_{kj} P\left(Y=y|X=x\right) = T_{ij} \tag{3}$$

First of all, we will use the classifier to training data and then classifier will be used on training data to estimate the transition matrix. Best estimated transition matrix will be selected based on test data accuracy.

3.2 Forward learning

Forward learning is a method that uses Transition Matrix (T in the formula) to adjust the loss of all training samples, thereby effectively reducing the negative impact of noise labels. Forward learning not only separates the teaching of transition probability from the teaching of the model but also adds an adaptive layer. This method modifies the loss of each sample by multiplying the estimated label conversion probability by the output of the specified deep neural network. It uses a linear combination of the softmax output of the deep neural network before applying the loss function. According to the formula shown below, after the noise adaptation layer, $P(Y|X)$ is flipped to $P(\widetilde{Y}|X)$.

$$[\,P(\widetilde{Y}=1|X),\cdots,P(\widetilde{Y}=C|X)]^T = T[P(Y=1|X),\cdots,P(Y=C|X)]^T \tag{4}$$

3.3 Backward learning

Backward learning initially used the softmax output of a deep neural network without loss correction to approximate the label transition matrix. According to the formula, contrary to Forward learning, Backward learning multiplies the predicted value by T^{-1}.

$$[P(Y=1|X),\cdots,P(Y=C|X)]^{T}=T^{-1}[P(\widetilde{Y}=1|X),\cdots,P(\widetilde{Y}=C|X)]^{T} \tag{5}$$

Commonly, when the transition matrix is known, Forward learning and Backward learning can be directly applied. If not, we need to build a transition matrix estimator provided in this article to estimate the transition matrix.

3.4 Cross Validation

Cross-validation is a data-related method whose function is to estimate the prediction error of the fitted model or training algorithm. Its necessary step is to divide the data into two parts, called training set and test set. The training set is used to fit the model or the training algorithm, and the test set is used to verify the performance of the fitting model or the training algorithm of the prediction model. When the data size is large enough, the typical proportion of the training set may be about 1/2 or 1/3.

In this assignment, we randomly select 80 percent training data as a training set, and then repeat the process and check the mean and standard deviation.

3.5 CNN

When we deal with heavy computer vision work such as image classification and label noise classification, the convolutional neural network comes to our mind because CNN is heavily used in these areas. This deep learning algorithm can take in an input image, assign importance (learnable parameters like weights, biases, etc.) to various aspects in the image, and be able to differentiate one image from the other. And the pre-processing required in CNN is much lower compared to other classification algorithms. In a nutshell, CNN can extract the feature of an image and convert it into a lower dimension without losing its characteristics.

There are six layers in CNN [1]:

1. Input layer

The input layer contains the image data, the image data should be represented by a three-dimensional matrix. For example, in the FashionMNIST0.5 dataset, we have an image shape of 28*28=784, we need to convert it to 28*28*1(784*1) before inputting it.

2. Convo layer

In the Convo layer, the features of the image are get extracted in this layer. The objective of the Convo operation is to extract the high-level features from the input image. Then, the first Convo layer is responsible for getting the low-level features, and with more layers be added, the level of the features will from low to high. The output will be the input for the next layer. In our project, it contains ReLU activation to make all negative values to zero.

3. Pooling layer

The pooling layer is used to reduce the spatial size of the convolved feature. It will decrease the computational power required to process the data through dimensional reduction. In this assignment, we are using the max-pooling method, which returns the maximum value from the portion of the image covered by the Kernel. It will better perform the de-noising along with dimensional reduction.

4. Fully connected layer

A fully connected layer involves weights, biases, and neurons. It will connect the neurons in one layer to neurons into another layer and use to classify images between different categories by training.

5. SoftMax layer

This layer is the last layer of CNN, the SoftMax is for multi-classification.

6. Output layer

The output layer contains the one-hot encoded label.

3.6 VGG

VGG is also a CNN model but performs much better than the traditional CNN model. It is considered to be one of the excellent vision models till now. The most important feature of the VGG model is that instead of having a very large number of hyper-parameters, it has the convolution layers of 3*3 filter with a stride 1 and always used the same padding with maxpooling layers of 2*2 filter of stride 2. It follows this setting of convolution layers and maxpooling layers consistently throughout the whole architecture. It will end with 2 FC(Fully connected) layers followed by a SoftMax output layer [2]. The architecture of VGG is shown in Figure 1

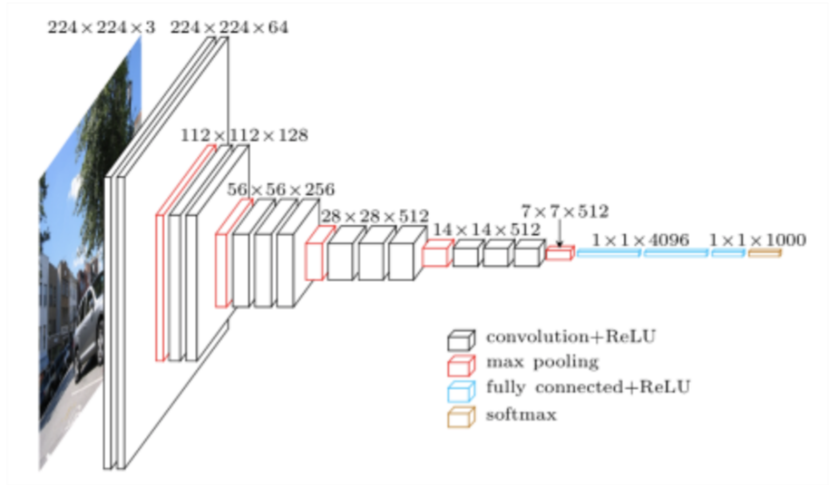
$224 \times 224 \times 3$
$224 \times 224 \times 64$
$112 \times 112 \times 128$
$56 \times 56 \times 256$
$28 \times 28 \times 512$
$14 \times 14 \times 512$
$7 \times 7 \times 512$
$1 \times 1 \times 4096$
$1 \times 1 \times 1000$
convolution+ReLU
max pooling
fully connected+ReLU
softmax

Figure 1: Architecture of VGG

3.7 Classifier Architecture

For all 3 data sets, we applied one CNN architecture and one VGG architecture. The architecture of the convolutional neural network consists of 2 blocks and each block contains 1 convolutional layer and 1 Max-Pooling layer, with the end of a dropdout layer and a FC layer. The RELU activation functions are utilized in the first N-1 layers and the softmax function in the last layer. The detail of each layer is shown in Table 1.

Table 1: CNN architecture

Layer	Input Size	Kernel Size	Feature Map
Input	-	-	-
Conv2D	28×28×1/32×32×3	3×3	32
Max-Pooling	28×28×1/32×32×3	2×2	32
Conv2D	28×28×1/32×32×3	3×3	64
Max-Pooling	28×28×1/32×32×3	2×2	64
Dropout	14×14×64/16×16×64	drop out rate 0.8	64
FC	128	-	3

The architecture of the VGG consists of 3 blocks and each block contains 2 convolutional layer, 1 Max-Pooling layer and 1 dropout layer, with the end of 2 FC layers. The RELU activation functions are utilized in the first N-1 layers and the softmax function in the last layer.The detail of each layer is shown in Table 2.

Table 2: VGG architecture

Layer	**Input Size**	**Kernel Size**	**Feature Map**
Input	-	-	-
Conv2D	28×28×1/32×32×3	3×3	32
Conv2D	28×28×1/32×32×3	3×3	32
Max-Pooling	28×28×1/32×32×3	2×2	32
Dropout	28×28×1/32×32×3	dropout rate 0.25	32
Conv2D	14×14×64/16×16×64	3×3	64
Conv2D	14×14×64/16×16×64	3×3	64
Max-Pooling	28×28×1/32×32×3	2×2	64
Dropout	14×14×64/16×16×64	dropout rate 0.25	64
Conv2D	14×14×64/16×16×64	3×3	64
Conv2D	14×14×64/16×16×64	3×3	64
Max-Pooling	14×14×64/16×16×64	2×2	64
Dropout	14×14×64/16×16×64	dropout rate 0.25	64
FC	512	-	128
FC	512	-	3

4 Experiment

4.1 Overview

In experiment, we used two different classifiers, CNN and VGG, with two different label noise robustness methods, forward and backward, on three datasets which have different shape and samples. Besides, we build an estimator to calculate transition matrix and compare result of estimated transition matrix with result of given transition matrix.

4.2 Dataset

4.2.1 FashionMINIST 0.5

There are 18000 samples in training datasets and the shape of training data is (n, images_shape). Each image shape is (28,28). Test dataset contains 3000 clean samples. Before training, training data will be split into 80 percent training set and 20 percent validation set.

The transition matrix of FashionMINIST 0.5 is:

$$\begin{bmatrix} 0.5 & 0.2 & 0.3 \\ 0.3 & 0.5 & 0.2 \\ 0.2 & 0.3 & 0.5 \end{bmatrix} \tag{6}$$

4.2.2 FashionMINIST 0.6

There are 18000 samples in training datasets and the shape of training data is (n, images_shape). Each image shape is (28,28). Test dataset contains 3000 clean samples. Before training, training data will be split into 80 percent training set and 20 percent validation set.

The transition matrix of FashionMINIST 0.6 is:

$$\begin{bmatrix} 0.4 & 0.3 & 0.3 \\ 0.3 & 0.4 & 0.3 \\ 0.3 & 0.3 & 0.4 \end{bmatrix} \tag{7}$$

4.2.3 CIFAR

There are 15000 samples in training datasets and the shape of training data is (n, images_shape). Each image shape is (32,32,3). Test dataset contains 3000 clean samples. Before training, training data will be split into 80 percent training set and 20 percent validation set.

4.3 Setup

For VGG classifier, the code run 5 times, and, in each time, the epoch is 15 and batch size is 128. To have convincing results, the CNN classifier uses the same parameters that loss function is categorical cross entropy and optimizer is rmsprop.

For CNN classifier, the code run 5 times, and, in each time, the epoch is 15 and batch size is 128. To have convincing results, the CNN classifier uses the same parameters that loss function is categorical cross entropy and optimizer is rmsprop.

4.4 Experimental results

Table 3: Experiment Results

Method	FashionMINIST 0.5	FashionMINIST 0.6	CIFAR
CNN	92.57%±0.4%	89.26%±0.9%	62.53%±3%
CNN with given TM	93.20%±0.1%	89.26%±0.9%	N/A
CNN with Estimated TM	93.13%±0.1%	89.28%±0.7%	67.23%±5.8%
VGG	91.87%±1.71%	87.35%±1.3%	61.49%±0.9%
VGG with given TM	92.48%±1.21%	87.35%±1.3%	N/A
VGG with Estimated TM	92.51%±1.16%	87.41%±1.37%	68.71%±2.68%

Table 4: Estimated Transition Matrix

Method	FashionMINIST 0.5	FashionMINIST 0.6	CIFAR
CNN	$\begin{bmatrix} 0.4903 & 0.1930 & 0.3167 \\ 0.2983 & 0.4745 & 0.2272 \\ 0.2240 & 0.2870 & 0.4890 \end{bmatrix}$	$\begin{bmatrix} 0.3657 & 0.2795 & 0.3548 \\ 0.2725 & 0.3832 & 0.3443 \\ 0.2695 & 0.2832 & 0.4473 \end{bmatrix}$	$\begin{bmatrix} 0.1806 & 0.3646 & 0.4548 \\ 0.1402 & 0.4704 & 0.3894 \\ 0.1242 & 0.3474 & 0.5284 \end{bmatrix}$
VGG	$\begin{bmatrix} 0.5268 & 0.1582 & 0.3150 \\ 0.3200 & 0.4210 & 0.2590 \\ 0.2248 & 0.2413 & 0.5338 \end{bmatrix}$	$\begin{bmatrix} 0.2880 & 0.2755 & 0.4365 \\ 0.2075 & 0.3773 & 0.4152 \\ 0.2040 & 0.2763 & 0.5338 \end{bmatrix}$	$\begin{bmatrix} 0.4304 & 0.0782 & 0.4914 \\ 0.4074 & 0.1318 & 0.4608 \\ 0.3176 & 0.0914 & 0.5910 \end{bmatrix}$

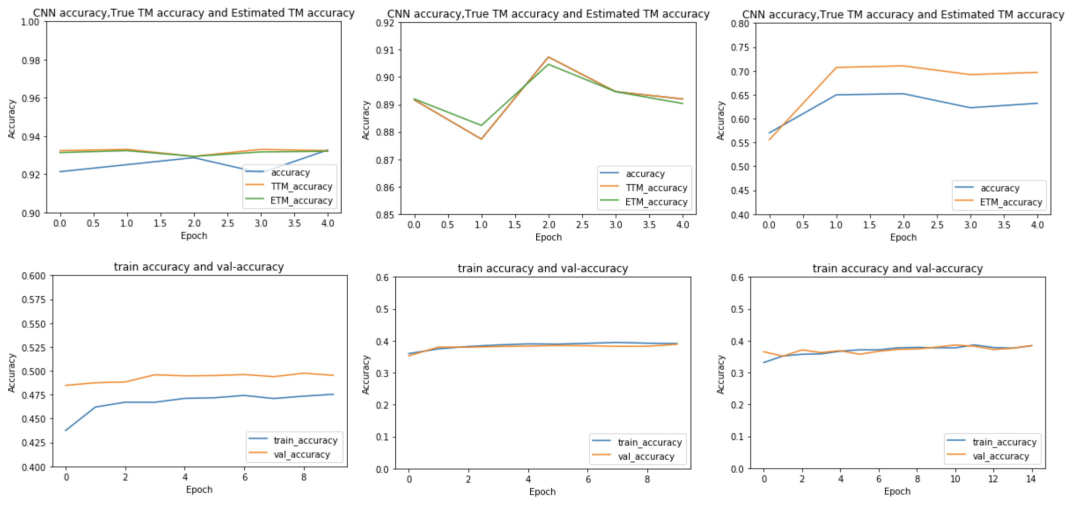

Figure 2: CNN graph results from left to right: Fashion0.5, Fashion0.6, Cifar

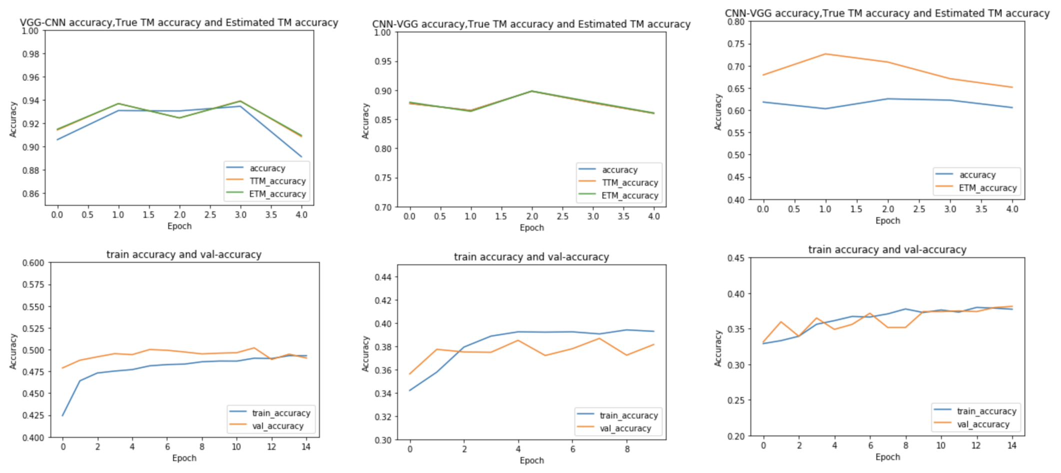

Figure 3: VGG graph results from left to right: Fashion0.5, Fashion0.6, Cifar

From Table 3 we can see that the FashionMINIST 0.5 dataset has the highest test accuracy, the FashionMINIST 0.6 dataset has the middle accuracy, and the CIFAR dataset has the lowest accuracy.

The accuracy of FashionMINIST 0.6 is not as good as FashionMINIST 0.5 because the accuracy of the training data of FashionMINIST 0.5 is about 50 percent, while the training data of FashionMINIST 0.6 accuracy is about 40 percent. Therefore, the learning effect of classifiers for FashionMINIST 0.5 is better, so the test accuracy is higher. The CIFAR test accuracy is lower because each image of CIFAR has 3 channels while each image of MNIST only has one channel.

The above results show that CNN has a better processing effect on the first two datasets. CNN is better than VGG in terms of accuracy and classifier stability. Although on the CIFAR dataset, CNN performs better than VGG, the performance of VGG with matrix is better than CNN with matrix, and at the same time, the stability of VGG is better. These two classifiers are relatively better for one-channel data, but they are relatively poor for three-channels data. For the transition matrix estimator(shown in Table 4), its performance is better than the given matrix because the test results of classifiers with estimator matrix have been improved, especially for the processing of CIFAR data, the accuracy is improved obviously.

5 Conclusion and Future work

In this assignment, we used two different classifiers with categorical cross-entropy loss function and rmsprop optimizer on three image datasets. However, the classifiers still have some weakness for label noise. Based on the above results and diagrams, in the future, we need to adjust the parameters of the classifiers, because in the process, the results of CNN for CIFAR show that the CNN model is not stable, and the deviation is relatively large. Secondly, the two classifiers are processing result of CIFAR's label noise is relatively low. Adjust the parameters or adjust the internal structure of the two classifiers, such as adding more layers to VGG, to enhance their learning ability.

References

[1] Dshahid380. (2019, February 26). Convolutional Neural Network. Retrieved from https://towardsdatascience.com/covolutional-neural-network-cb0883dd6529

[2] Thakur, R. (2019, August 20). Step by step VGG16 implementation in Keras for beginners. Retrieved from https://towardsdatascience.com/step-by-step-vgg16-implementation-in-keras-for-beginners-a833c686ae6c

[3] Yang, B. (2019). On the robust splitting criterion of random forest. *Proceedings - IEEE International Conference on Data Mining, ICDM,* 2019-, 1420–1425. https://doi.org/10.1109/ICDM.2019.00184

[4] Zhou, X. (2019). Improving Robustness of Random Forest Under Label Noise. *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 950–958. https://doi.org/10.1109/WACV.2019.00106

Appendix

Contributions of team members

The code is written in cooperation, and each person is responsible for different parts of the code, as well as data collection and analysis. After finishing all the coding work, we used Overleaf to write the report together. In short, each team member made the same contribution.

Code environment

Hardware and operation system information

Operation system: macOS 10.15.6

CPU: i7-8700HQ

GPU: Intel UHD Graphics 630

RAM: 16GB

SSD: 512GB

Python environment

python 3.7.4

numpy 1.15.1

tensorflow 2.2.0

matplotlib 3.1.1

How to run the code

Run method

Place the data set files and code files in the same directory, open each code file(.ipynb file) in jupyter notebook using terminal. Load the dataset first and run each code block step by step to get the result.

Tips

For the CNN classifier, open the ”COMP5328_Assignment2_CNN_Classifier.ipynb”. For the VGG classifier, open the ”COMP5328_Assignment2_VGG_Classifier.ipynb”.