Accuracy Improvement

Author:Yunxiang Zhu[1*] Yanpeng Zhao[2] Guanghong Liu[3] Lifan Zeng[4]

1.School of Mechanical and Electrical Engineering and Automation, Shanghai University, Shanghai, 200444, China

2.University of Wisconsin-Madison, Wisconsin, 53711, America

3.School of Electrical Engineering, Zhejiang University, Zhejiang, 310027, China

4. Electrical engineering of The State University Of Buffalo, buffalo, NY, 14260, America

Adviser: Professor Roman Kuc

Abstract

Since AlphaGo beat the world Go champion in 2016, which attracted wide attention, the neural network has become more and more popular in recent years, and people's research on it has gradually improved and been applied in different fields. Today, artificial intelligence and machine learning have become an essential part of modern society and intelligent systems. We do image recognition, speech recognition, and visual learning, closely related to machine learning.

However, in machine learning, unsatisfactory training results or even training failure is always encountered. Therefore, in machine learning, it is imperative to improve the accuracy of neural network training results. In this paper, speech recognition, image processing, MNIST, and other classical neural network models will be used to set the training parameters of the neural network better and improve the accuracy of training through voting、quantization、restart and other methods.

The part of research is aiming to find the relationship between restart numbers on the training process and the total extent of learning improvement. At the same time, several algorithms on utilizing these restart numbers are to be compared and selected. Finally, the conclusion is drawn that the more restart made in the training process with a convolutional neural network, the less profit on accuracy improvement we gain from restarting the process.

Keywords: accuracy improvement; machine learning; committee voting; restart; convolutional neural network; network model; stopping rule; MNIST database; speak recognition

1 Introduction

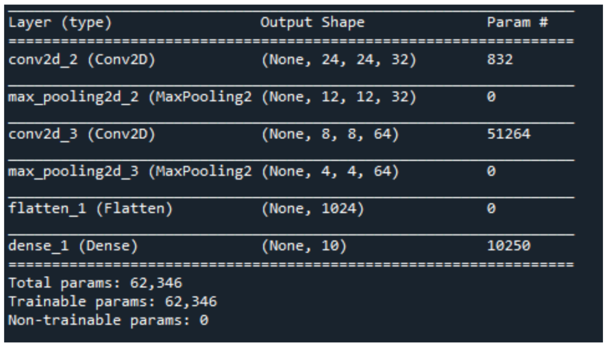
1.1 Overview

Artificial intelligence has been applied to more and more areas at the basement of hardware technology mature. Among artificial intelligence including PAC learning, RC learning, Online learning, Neural network or convolutional neural networking which is applied in this research is especially popular and being used in different classifying, predicting scenarios. The neural network let machines learn themselves through training the data set. Before training the data set, doing some processing and optimization on the data set is one of the methods to improve accuracy. Through experiments, the paper showed that multiple measurements through committee voting and restart after training also helped improve the accuracy of training.

1.2 CNN structure, Activation, Loss Function

1.2.2 Neural Network

For studying the relationship between learning profit and restart numbers, the work chose simply constructed convolutional neural networks. The CNN is constructed with two dimensional layers. The first layer is in shape 24*24*32 and using 32 5*5 filters. The shape is 24*24 because the input is reshaped to 28*28 pixels images, and filter is 5*5. Multiplying each 5*5 pixels sub-image in the original input makes the output of the first layer diminished by 4 on each side. 32 different filters make the first layer 32 different filtered 24*24 image. After the first convolutional 2-dimension layer, a max pooling which selects 1 of each 2*2 pixels in the first convolutional 2-dimensional layer output is added. Dimension of first pooling output is diminished to 1/4 which is 12*12*32. Same process is gone through in the second convolutional two-dimensional layer. The output of second convolutional layer becomes 8*8*64. Same pooling mechanism is utilized after the second convolutional layer. The dimension is further step down to 4*4*64 dimensions. Now 4*4*64 output is transformed into one dimensional linear 1024 inputs for a multilayer perceptron with activation function of SoftMax.(shown after figure 1) And the final classifying result should be a one-hot label with length 10 for there are in total 10 different classes. Fashion MNIST database (figure 2) is a dataset consisting of 60000 training images including labels, and 10000 test images accompanied with labels. The goal of this neural network is to classify those images recognizing for example which one is high heel shoes, which one trousers.



```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_2 (Conv2D)            (None, 24, 24, 32)        832
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 12, 12, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 8, 8, 64)          51264
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 4, 4, 64)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 1024)              0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                10250
=================================================================
Total params: 62,346
Trainable params: 62,346
Non-trainable params: 0
```

Figure 1: The verbal structure of CNN used for the research

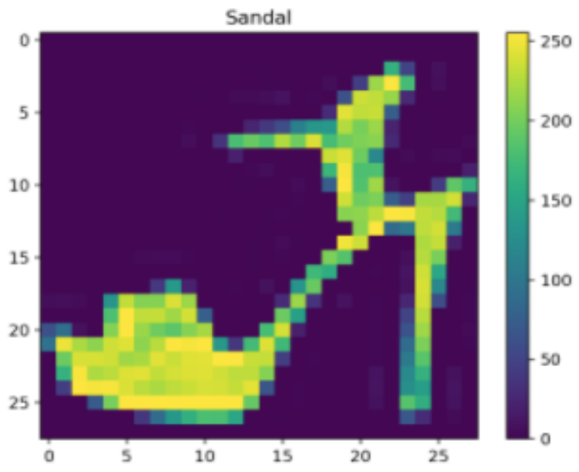


Figure 2: Fashion MNIST example

1.2.3 Activation, Loss Function

RELU which when the input is lower or equal to 0 is outputted as 0, or output is input itself is used in both 2 convolutional layers (Figure 3). And for the last multilayer perceptron, we use SoftMax to output one-hot result. Loss function is categorical cross entropy that is the sum of log predicted output times the real label.

Formula (1): $$Loss = -\sum_j d_j \log P_j$$

Formula (2): $$\text{SoftMax}: P_k = \frac{exp(z_k)}{\Sigma_i exp(z_i)}$$

Using chain rule, **Formula (3)** can be got: $\frac{\partial Loss}{\partial w_j} = \frac{\partial Loss}{\partial P_j} \cdot \frac{\partial P_i}{\partial X_j} \cdot \frac{\partial X_j}{\partial w_j}$

The Gradient of loss which is essential to observe global optima or local minima is:

$$\frac{\partial Loss}{\partial z_k} = \sum_t \frac{P_i}{t_i} * P_i P_k (i \neq k) + \frac{t_k}{P_k} P_k * (P_k - 1) = P_k - t_k$$

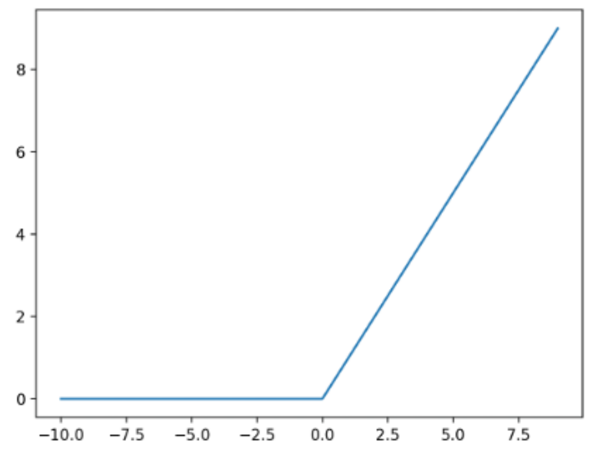


Figure 3: Relu plot

The accuracy improvement observation is due to the decrease of PE (acronym of percentage error) value of a trained model. When PE value decreases, the accuracy of an expert improves. Restart enables training process randomly to start at the point of function. This randomly restart is functioned to escape local minima and ideally can approach to global minima between predicted label and truth label. In this case, more times the training process restarts, the more possible we can get to global minimal. But the truth is, the improvement of possibility finding global minima decreases along with more restarts. In the rest part of this essay, ideal restart number and algorithm utilizing those restarts is going to be introduced. A function between restart number and accuracy, relationship between restart number and learning rate would also been drawn.

1.3 Deep neural network

This work also take the experiment of speech recognition as an example which use a typical deep neural network. Its principle is briefly introduced here. According to the position of different layers, the neural network layers inside DNN can be divided into three types: input layer, hidden layer, and output layer. Generally speaking, the first layer is the input layer, the last layer is the output layer, and the middle layers are all hidden layers.

The layers are fully connected; any neuron in layer I must be connected to any neuron in layer i +1. Although DNN looks complicated, it is still a linear relationship $Z=\Sigma\omega ixi+b$ with an activation function $\sigma(z)$. Since there are many layers of DNN, the number of our linear relationship weight coefficient ω and bias b is also significant. A common DNN model with n inputs,two hidden layers and 10 outputs can be seen in Figure 4.

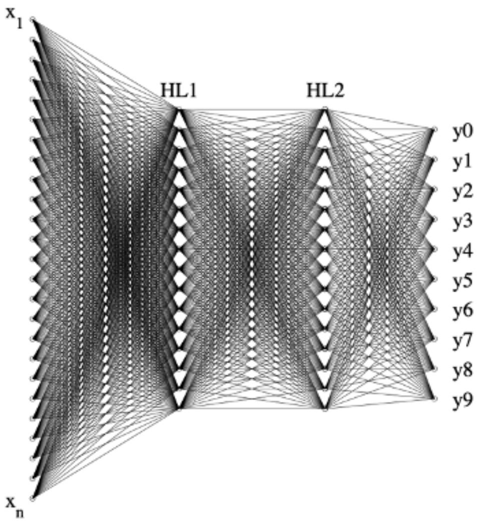


Figure 4:simple DNN model

In the training sample, X is the input vector, and the feature dimension is 28*28, while Y is the output vector and the feature dimension is 10. In this work a model need to be trained with these samples. Using the other half of the data to form a test sample of the same size, then the output of the vector can be predicted. At this time, the work will find the appropriate linear weight matrix(ω) and bias vector corresponding to all the hidden layers and output layers, namely the appropriate hyperparameter so that the output calculated by all the training sample inputs is as equal to or very close to the sample output as possible. At the same time, the test results should not differ too much from the training results.

An appropriate loss function is also used to measure the training samples' output loss and then optimize this loss function to find the minimum extreme value. The corresponding series of linear weight matrix ω and bias vector b constitute our optimal model. In DNN, the most common method to solve the optimal extremum of the loss function is generally completed step by step iteratively by the gradient descent method. The setting of hyperparameters such as ω, b and the optimization of backpropagation was learned in detail in previous lectures and will not be repeated here.

The scale of the input feature vector is so large that cause too much weight ω and bias b need to be considered and calculated, which not only makes the calculation time longer but also has a great influence on the training results of models with different activation functions and various model layers(make the model training results unstable). Therefore, it is very important to understand the storing capacity of the spectrogram and the training limit of the network model, which can make it easier to reduce and simplify the feature vector in the subsequent improvement of accuracy.

2 Methods

2.1 Voting

2.1.1 Voting

Voting is a kind of ensemble learning. Normally a stable and generally well-performed model must be prepared. However, what we get are amount of models with preference in some aspects. In this case, ensemble learning combine these models expecting to get a better model. Sometimes one model makes a mistake, other models would correct the mistake and output a correct answer.

There are two kinds of voting, hard voting and soft voting. Hard voting is shown in figure 5, which means every model gives their votes and the class having the most votes is taken as the final classification. Soft voting means to calculate the average probability of each class and compare the averages. The class having the biggest average probability is taken as the final classification.

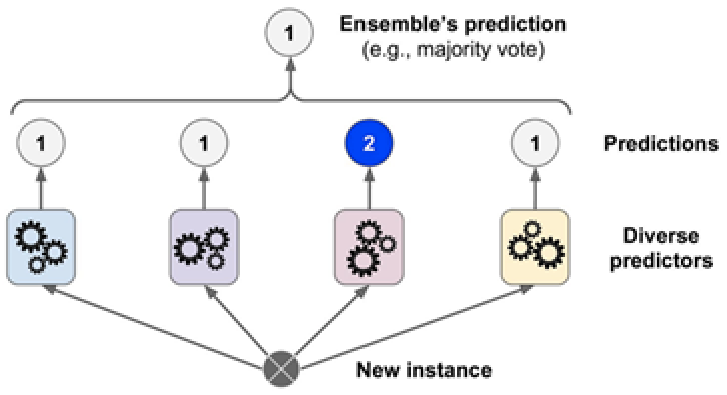


Figure 5: Hard voting

2.1.2 Get more models

Since the goal is to look into the impact of different number of models, the first step is to get enough models. Restart is a good method to get new models. In training process, at first, we need some initial weights and bias, and then, we train the model by updating weights and bias times and times again to get lower loss and higher accuracy. Usually initial weights and bias are randomly defined, for which the results are different every time we restart.

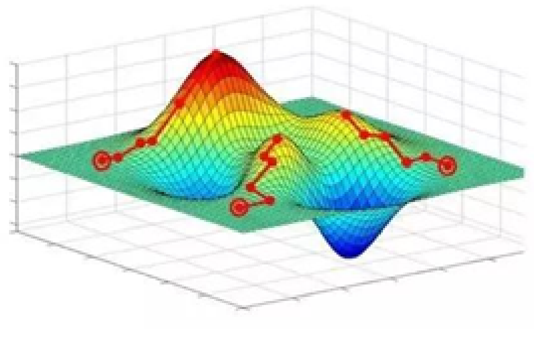


Figure 6: Simple example of loss as function of weights

As Figure 6 shows, with different initial values, paths are different, so results are different.

Apart from getting more models, to improve accuracy, some bad models should be eliminated. Therefore, a threshold should be determined. The work use hyperparameters determined and restart for 100 times to generate Figure 7.

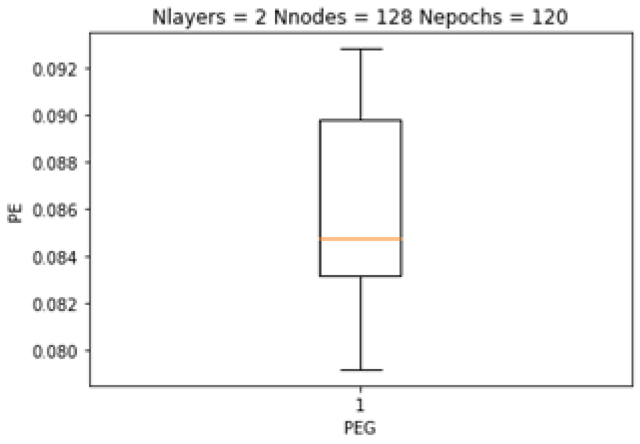


Figure 7: Nrestart = 100 boxplot

According to Figure 7, the bottom of the square is PEG = 0.083, so the threshold = 0.083. Then continue restarting training to see if the PEG value<0.083,which means the model is qualified. In this way, the work get 39 models. Finally, using hard voting for 6 times, the results are shown in section 4.

2.2 Quantization of dataset

The work mainly aims to observe and study the storing capacity of the neural network and spectrogram to the input feature matrix to provide help and preparation for adopting methods to improve accuracy. After understanding the limitations of data conversion to the feature matrix, people can choose the originally random weights of the data set relatively accurately. Meanwhile, the data dimension might be reduced by pooling or other methods to improve the speed and accuracy of the result. There is no doubt that there are some processes on the data before training in the neural network, but the scale of the data is adjustable. It affects the ranking of our training and the accuracy of the training. In the neural network structure, this work take speech recognition as the entry point of observation and research. The paper choose authors' recording data as the data set, Softmax (Relu) as activation functions, and gradient descent method as the optimization methods. In data processing, the result shows that the normalization of input features can significantly improve the effect and accuracy of training. After normalization, the quantization of input features data by various bits also affects the data storing ability of neural networks and spectrogram. In the real-time classification, the training result of the neural network reach the expected degree after quantizing by 7 digits, that is, 128 threshold classification and above.

2.2.1 the impact of dataset size on accuracy

The scale of input feature vector is so large that cause too much weight ω and bias b need to be considered and calculated, which not only makes the calculation time longer, but also has a great influence on the training results due to different activation functions and model layers(make the model training results unstable).Therefore, it is very important to understand the storing capacity of spectrogram and the training limit of the network model, which can make it easier to reduce and simplify the feature vector in the subsequent improvement of accuracy.The following paper takes the experiment of speech recognition as an example.

2.2.2 Turn our sound into bits signal

After the voice is recorded, they should be put into the computer, but sound travels in the form of waves. Sound waves are one-dimensional. At each moment, a height value is tested basing on the height of the wave. To convert this sound wave to a number, the work record the height of the wave at equally spaced points: the work take thousands of readings per second and record the number representing the height of the sound wave then. For speech recognition, a sampling rate of 16khz (16,000 samples per second) is sufficient to cover the frequency range of human speech. In this experiment, the sample rate of 8000 is set for 3 seconds. At the same time, the principle of random sampling also shows how to quantize the recording data set skillfully.

The figure 8 shows the converted signal diagram after the English zero recording.

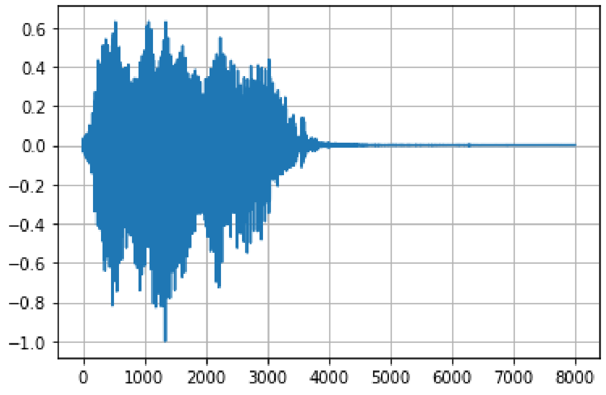


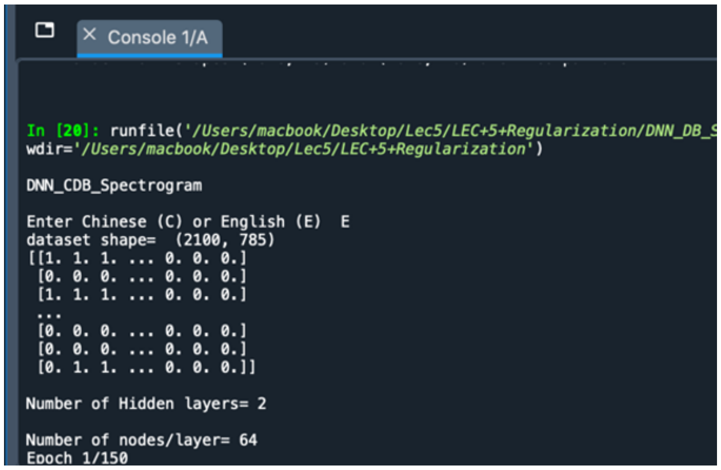
Figure 8: recording signal(zero in English)

2.2.3 The algorithm steps

First of all, the work set a threshold value and traverse the NumPy number groups for judgment. When the input data is more significant than this value, it goes to 1, and vice versa. As the figure 5 shows, the input features are completely divided into 0 and 1. They were repeatedly trained and tested as a training set, and the final PEG was between 0.57 and 0.22

Then the work multiply the spectrogram by n bits, which is range from 2 to 8. After that, the integer value is taken and divided by those digits. Finally, this work applied the spectrogram to the neural network to observe the changes in PEG values

For example, in the case of the converted input characteristics being quantized by 2 bits, the converted input feature matrix is shown in the following figure(figure 9), which is equivalent to setting a threshold to discretize the data set.



```
In [20]: runfile('/Users/macbook/Desktop/Lec5/LEC+5+Regularization/DNN_DB_S
wdir='/Users/macbook/Desktop/Lec5/LEC+5+Regularization')

DNN_CDB_Spectrogram

Enter Chinese (C) or English (E)  E
dataset shape=  (2100, 785)
[[1. 1. 1. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [1. 1. 1. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 1. 1. ... 0. 0. 0.]]

Number of Hidden layers= 2

Number of nodes/layer= 64
Epoch 1/150
```

Figure 9: The input data set after the threshold is set

2.3 Restart Algorithm and stopping rules

2.3.1 Simple version algorithm

It is not easy to acquire a perfect way to get global minima in the loss function, which reaches the highest accuracy. There are some optimizers that enable us to jump out of local minima in search of global minima. Among them, an optimizer called Adam is relatively more effective and more popular compared to other optimizer functions like SGD or Gradient descendent. Adam automatically adjusts the learning rate for each weight. But powerful optimizer at the same time weakens the usefulness of restart itself. Directly gradient descent might be a better way to understanding how restart improves the accuracy, but not many people applying gradient descent as an optimizer in their neural network model. Gradient descent is the easiest way to adjust different weights and is very easy to understand and implement. The problem is it can be stuck at local minima or saddle points. Gradient descendent also require memory to compute derivatives of the entire dataset. Such a terrible optimizer does not worth research in this case. The algorithm for best-utilizing restart is written below:

2.3.1.1 Algorithm steps

1)Input our desired low PE value for the trained model
2)Have ten restarts on the model with 13 epochs each
3)If single restart reaches the PE value we want, the iteration is jumped out
4)Else it selects the best or lowest PE value among ten restarts

The work want to observe the relation between profit gain of accuracy and restart number so the restart number doesn't have to be exactly ten. The conjecture (Figure 10) is that the more restart number this work have on the trained model, the lower profit gained from the learning. But the accuracy inevitably goes higher accompanied by restart number. A function between accuracy improved and restart number is shown in figure (Figure 11). Now taking the derivative or gradient of the curve and then the value finally converges to zero as the restart number goes high.

There are smarter and more directly related to loss function local minima restart algorithms that existed in the artificial intelligence area.Ten restart is not enough to see salient discrepancies between restarts.Once the number goes higher, we can see more significant improvement.(Figure 12)

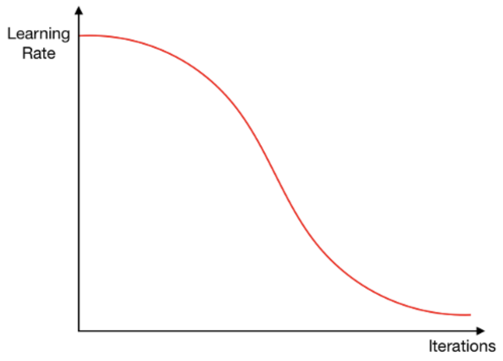


Figure 10: The conjecture on restart number and learning profit

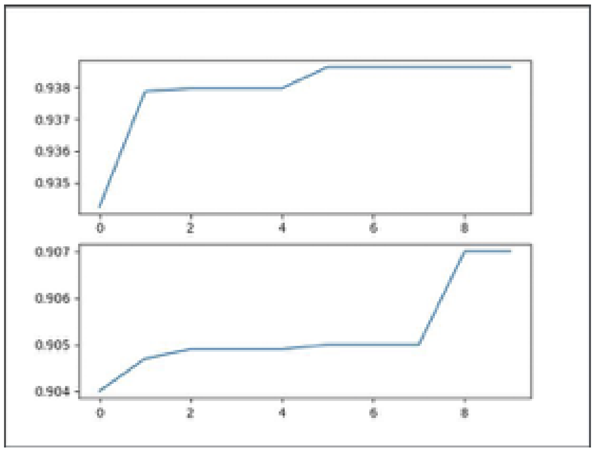


Figure 11: The above is relationship between train set accuracy and restart below is relationship between test set accuracy and restart

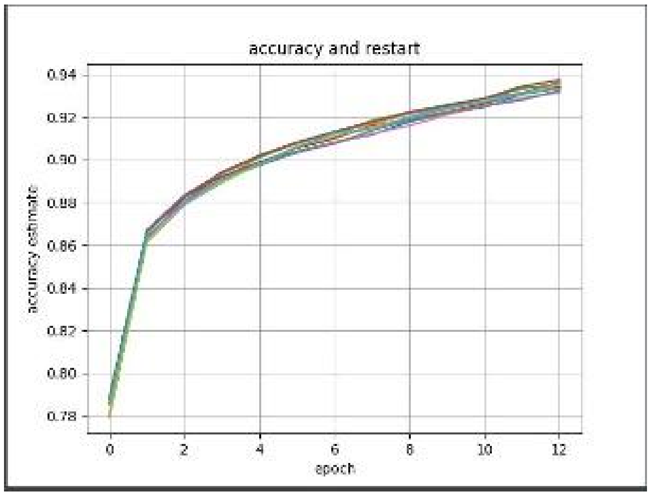


Figure 12: The performance of 10 restarts

2.3.2 Bayesian stopping rules

The Bayesian stopping method consists of three parts that are prior function, likelihood function, and loss function that premised in every restart algorithm. The likelihood relates the unknown true discrete distribution over local optima to the observed quantities. In combination with prior function on the unknowns, the posterior probability to unknown true distribution can pertain. And this algorithm stops whenever the expected loss under the current posterior would increase with further starts.[6]

In conclusion in the equation:

Present Uncertainty = Past Uncertainty * predictive updating factor

That:

$$\text{Formula (3):}\underbrace{P(\theta|data)}_{\text{Posterior}} = \underbrace{P(\theta)}_{\text{prior}} * \underbrace{\frac{P(data|\theta)}{P(data)}}_{\text{predictive}}$$

2.3.3 Heuristic stopping rule

The mechanism of the heuristic stopping rule is that there are finite minima on the range of function. They stop once a new minimum has not been found through V restart. The number of restarts allowed to find the next minimum depends on the convergence rate of the sample variance of different statistics, and usually increases with the number of minimum values found.[6]

2.3.4 Dorea's stopping rule

This rule can estimate the probability that the function value of one local optimal is in the range of global optimal. And there are two stopping conditions: it stops at the same time the probability of the local optima is in the range of global optimal is high enough. Second, the iteration stops when there's no improvement on the last several steps (defined by the user).[6]

Among these three stopping rules, Dorea's stopping rule usually stops below ten restarts while other stopping rule mentions pass ten, and one rule utilizing Bayesian stopping rule even uses 10^2 to reach global minima. The average of those different restart rules on finding minimum function value is shown by *Dick et al.*, And it is a coincidence with our research figure output. As figure 13 shows, taking the derivative of the averaged line in 8000 trials, the figure should also converge to zero as the restart number approaches infinity.[5]

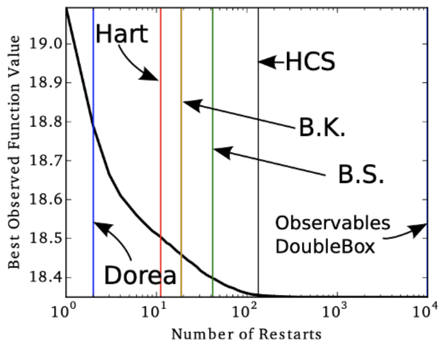


Figure 13: Best Observed Function Value-Number of Restarts

3 Results

3.1 Voting

PE (Probability of Error) is usually used to evaluate the performance of a model. After six times running, different PE of different numbers of models are shown. Since voting needs an odd number of models, lists show PE of 1, 3, 5, … , 39 models voting.

```
PE1 = [0.0592, 0.0564, 0.05493333, 0.0534, 0.05248, 0.0512,
0.04971429, 0.0486, 0.04737778, 0.04608, 0.04516364, 0.0464,
0.046, 0.046933333333333334, 0.0474, 0.048, 0.04906666666666667,
0.04994285714285714, 0.0506, 0.05102222222222222]

PE2 = [0.064,   0.0652,    0.06533333, 0.0652,   0.06464,
0.0636, 0.06217143, 0.0608,     0.05902222, 0.05752,    0.056,
     0.05506667, 0.05452308, 0.05422857, 0.05381333, 0.0537,
0.05369412, 0.05382222, 0.05389474, 0.05396]

PE3 =  [0.0688,      0.0684,     0.0664,   0.0654,    0.06464,
0.06346667, 0.0616,     0.0594,      0.05697778, 0.0548,
0.0528,    0.05126667, 0.04996923, 0.04908571, 0.04853333,
0.0484,    0.04832941, 0.04835556, 0.04837895, 0.04836]

PE4 = [0.0592,     0.0568,      0.05573333, 0.055,    0.05456,
0.05373333, 0.05257143, 0.051,       0.04924444, 0.04776,
0.04654545, 0.04593333, 0.04541538, 0.04531429, 0.04522667,
0.04545,    0.04574118, 0.04604444, 0.04635789, 0.0466]

PE5 = [0.0632,     0.0616,      0.0608,    0.06,       0.05936,
0.05813333, 0.05691429, 0.0553,      0.05324444, 0.05152,
0.05010909, 0.04933333, 0.04836923, 0.048,      0.04773333,
0.0478,    0.04795294, 0.04808889, 0.04821053, 0.04828]

PE6 = [0.06,       0.0584,      0.05653333, 0.0552,   0.05472,
0.05413333, 0.0528,   0.0515,     0.04977778, 0.04816,
0.04683636, 0.046, 0.04535385, 0.04508571, 0.04512,   0.04535,
0.04555294, 0.04582222, 0.04614737, 0.04644]
```

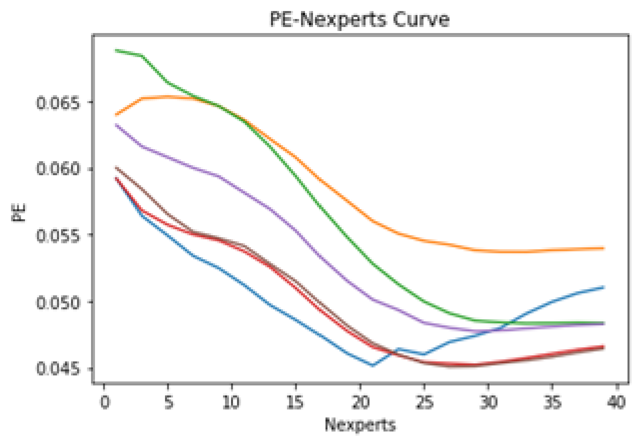


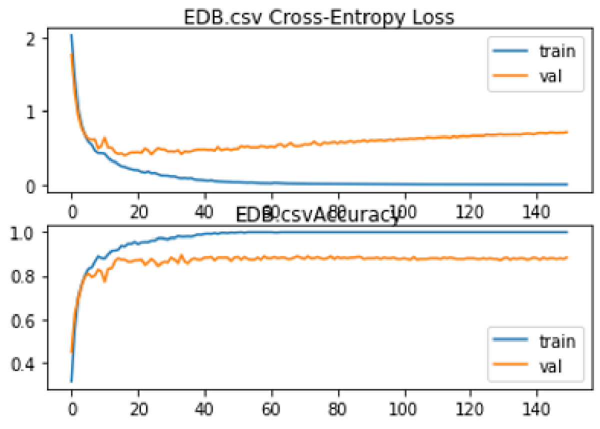
Figure 14: 6 times PE-Nexperts curve

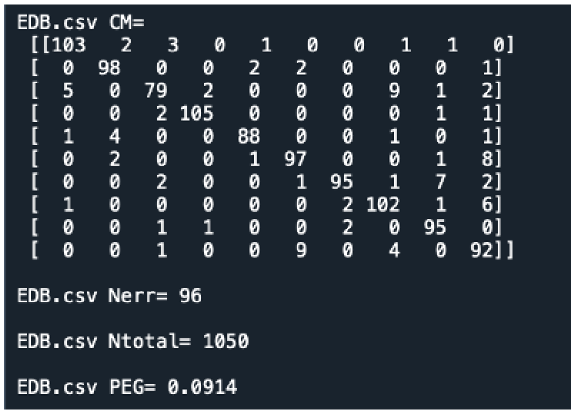
Figure 14 shows how PE value changes as the number of models is bigger.According to the curves, it is easy to get a conclusion that there is a minimum PE value, and it usually appears when Nexperts is bigger than 20 and smaller than 35. When Nexperts is bigger than 35, PE increases as Nexperts becomes bigger.

3.2 Quantization of data

The experimental results are represented by programming and automatically calculated to form figures. The horizontal axis in the left figure below represents the number of restarts, the blue line represents the loss function and accuracy during training; the orange line represents the loss function and accuracy during testing. The figure on the right shows the confusion matrix, the number of recognition errors and the total number of recognitions, and the average correct rate.

3.2.1 without quantization

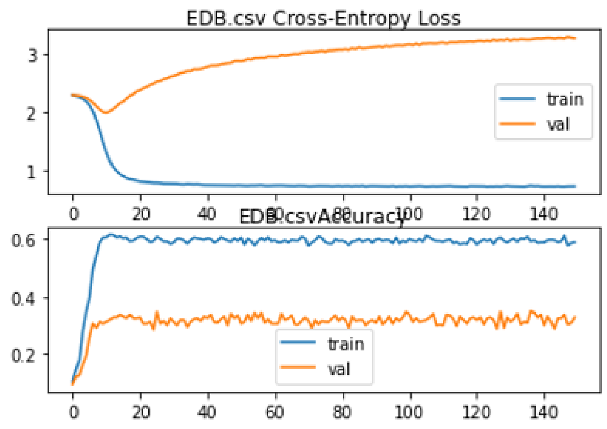


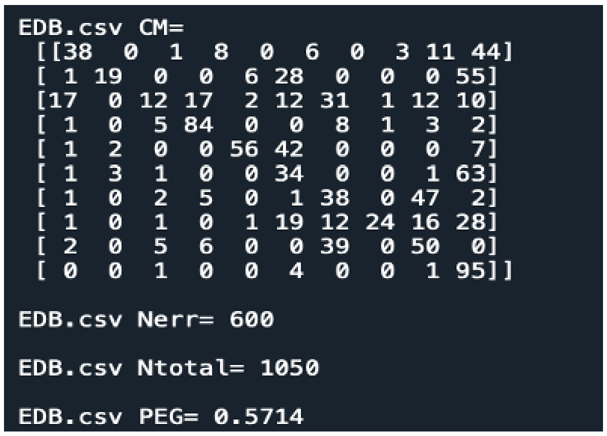


```
EDB.csv CM=
 [[103   2   3   0   1   0   0   1   1   0]
 [  0  98   0   0   2   2   0   0   0   1]
 [  5   0  79   2   0   0   0   9   1   2]
 [  0   0   2 105   0   0   0   0   1   1]
 [  1   4   0   0  88   0   0   1   0   1]
 [  0   2   0   0   1  97   0   0   1   8]
 [  0   0   2   0   0   1  95   1   7   2]
 [  1   0   0   0   0   0   2 102   1   6]
 [  0   0   1   1   0   0   2   0  95   0]
 [  0   0   1   0   0   9   0   4   0  92]]

EDB.csv Nerr= 96

EDB.csv Ntotal= 1050

EDB.csv PEG= 0.0914
```

Figure 15: PEG value and confusion matrix of original model

3.2.2 quantize by 1 bit



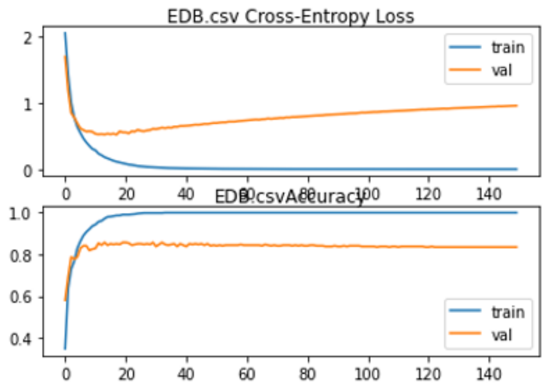


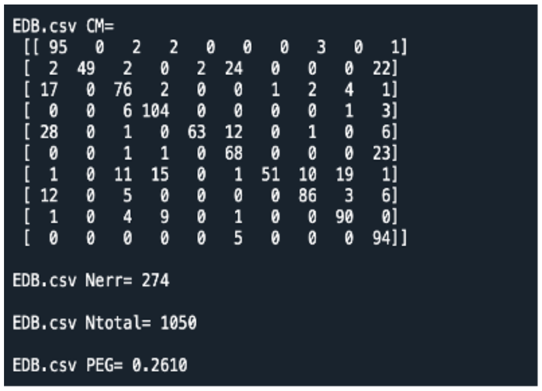
```
EDB.csv CM=
 [[38  0  1  8  0  6  0  3 11 44]
 [ 1 19  0  0  6 28  0  0  0 55]
 [17  0 12 17  2 12 31  1 12 10]
 [ 1  0  5 84  0  0  8  1  3  2]
 [ 1  2  0  0 56 42  0  0  0  7]
 [ 1  3  1  0  0 34  0  0  1 63]
 [ 1  0  2  5  0  1 38  0 47  2]
 [ 1  0  1  0  1 19 12 24 16 28]
 [ 2  0  5  6  0  0 39  0 50  0]
 [ 0  0  1  0  0  4  0  0  1 95]]

EDB.csv Nerr= 600

EDB.csv Ntotal= 1050

EDB.csv PEG= 0.5714
```

Figure 16: PEG value and confusion matrix after quantizing by one bit

Error rate=525.16%

3.2.3 quantize by 2 bits



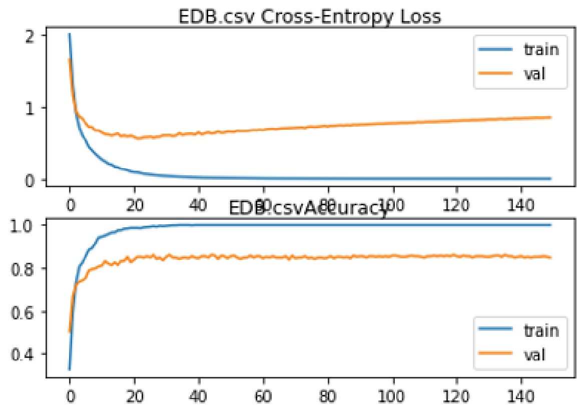


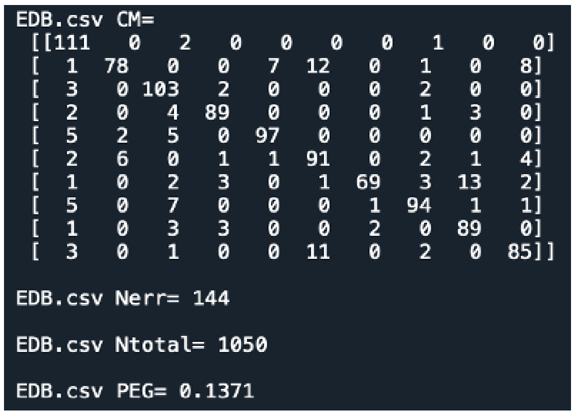
```
EDB.csv CM=
 [[ 95   0   2   2   0   0   0   3   0   1]
 [  2  49   2   0   2  24   0   0   0  22]
 [ 17   0  76   2   0   0   1   2   4   1]
 [  0   0   6 104   0   0   0   0   1   3]
 [ 28   0   1   0  63  12   0   1   0   6]
 [  0   0   1   1   0  68   0   0   0  23]
 [  1   0  11  15   0   1  51  10  19   1]
 [ 12   0   5   0   0   0   0  86   3   6]
 [  1   0   4   9   0   1   0   0  90   0]
 [  0   0   0   0   0   5   0   0   0  94]]

EDB.csv Nerr= 274

EDB.csv Ntotal= 1050

EDB.csv PEG= 0.2610
```

Figure 17: PEG value and confusion matrix after quantizing by two bits

Error rate==185.56%

3.2.4 quantize by 3 bits



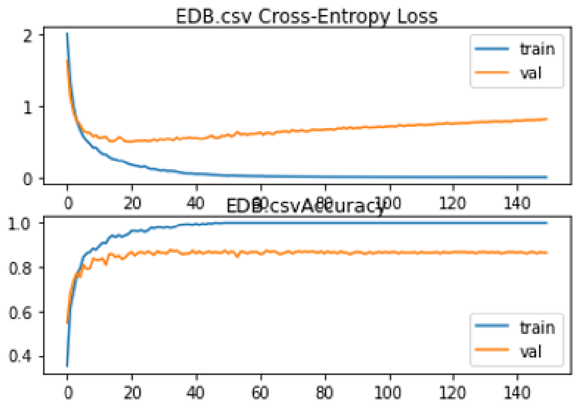


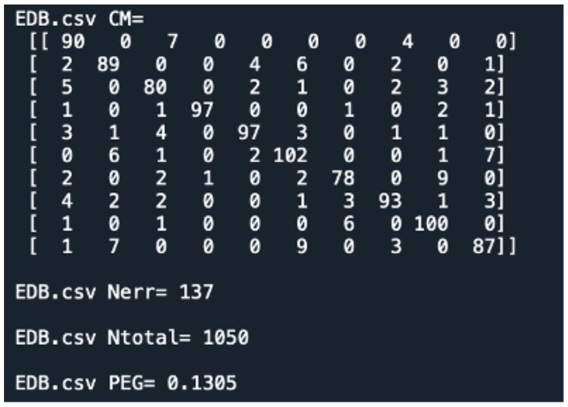
```
EDB.csv CM=
 [[111   0   2   0   0   0   0   1   0   0]
 [  1  78   0   0   7  12   0   1   0   8]
 [  3   0 103   2   0   0   0   2   0   0]
 [  2   0   4  89   0   0   0   1   3   0]
 [  5   2   5   0  97   0   0   0   0   0]
 [  2   6   0   1   1  91   0   2   1   4]
 [  1   0   2   3   0   1  69   3  13   2]
 [  5   0   7   0   0   0   1  94   1   1]
 [  1   0   3   3   0   0   2   0  89   0]
 [  3   0   1   0   0  11   0   2   0  85]]

EDB.csv Nerr= 144

EDB.csv Ntotal= 1050

EDB.csv PEG= 0.1371
```

Figure 18: PEG value and confusion matrix after quantizing by three bits

Error rate=50%

3.2.5 quantize by 4 bits



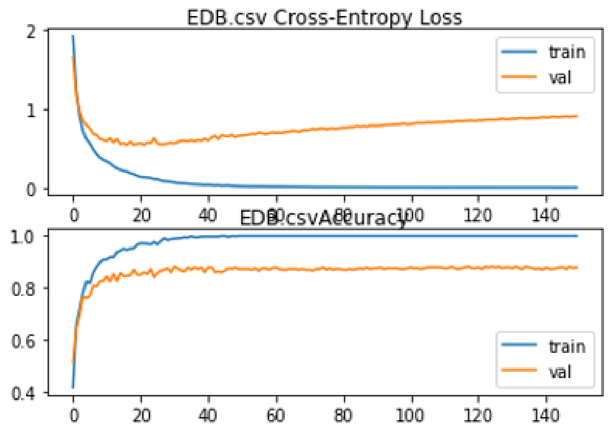


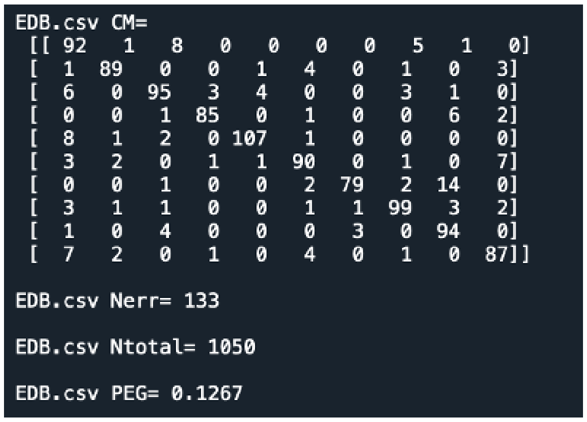
```
EDB.csv CM=
 [[ 90   0   7   0   0   0   0   4   0   0]
 [  2  89   0   0   4   6   0   2   0   1]
 [  5   0  80   0   2   1   0   2   3   2]
 [  1   0   1  97   0   0   1   0   2   1]
 [  3   1   4   0  97   3   0   1   1   0]
 [  0   6   1   0   2 102   0   0   1   7]
 [  2   0   2   1   0   2  78   0   9   0]
 [  4   2   2   0   0   1   3  93   1   3]
 [  1   0   1   0   0   0   6   0 100   0]
 [  1   7   0   0   0   9   0   3   0  87]]

EDB.csv Nerr= 137

EDB.csv Ntotal= 1050

EDB.csv PEG= 0.1305
```

Figure 19: PEG value and confusion matrix after quantizing by four bits

Error rate=42.78%

3.2.6 quantize by 5 bits



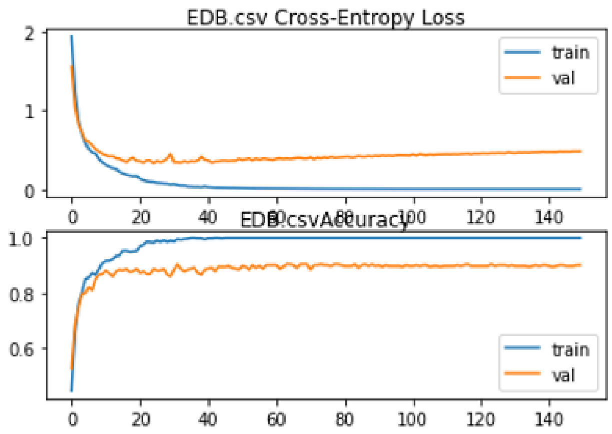


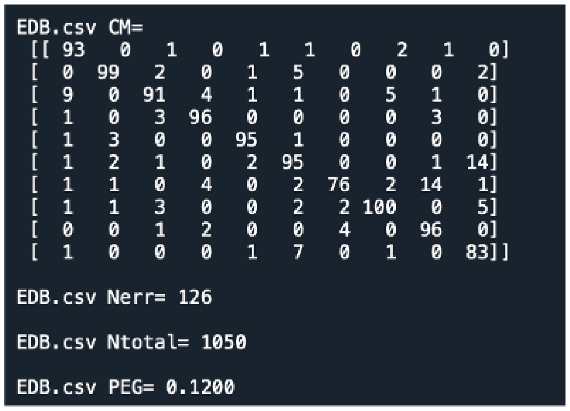
```
EDB.csv CM=
 [[ 92   1   8   0   0   0   0   5   1   0]
 [  1  89   0   0   1   4   0   1   0   3]
 [  6   0  95   3   4   0   0   3   1   0]
 [  0   0   1  85   0   1   0   0   6   2]
 [  8   1   2   0 107   1   0   0   0   0]
 [  3   2   0   1   1  90   0   1   0   7]
 [  0   0   1   0   0   2  79   2  14   0]
 [  3   1   1   0   0   1   1  99   3   2]
 [  1   0   4   0   0   0   3   0  94   0]
 [  7   2   0   1   0   4   0   1   0  87]]

EDB.csv Nerr= 133

EDB.csv Ntotal= 1050

EDB.csv PEG= 0.1267
```

Figure 20: PEG value and confusion matrix after quantizing by five bits

Error rate==38.62%

3.2.7 quantize by 6 bits



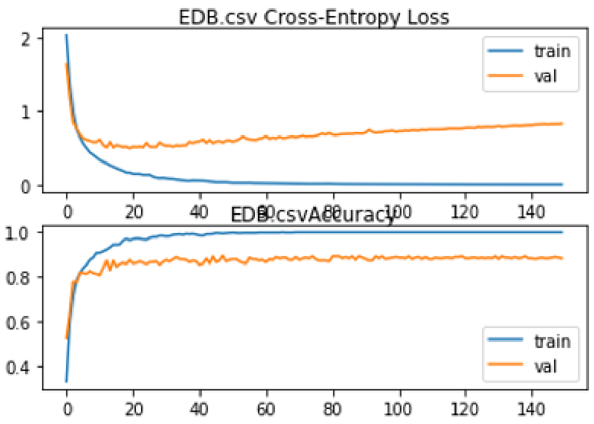


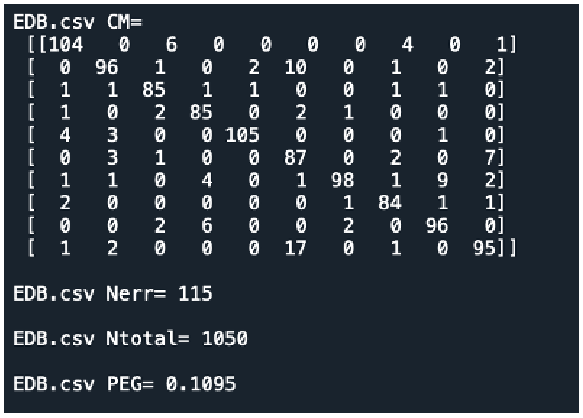
```
EDB.csv CM=
 [[ 93   0   1   0   1   1   0   2   1   0]
 [  0  99   2   0   1   5   0   0   0   2]
 [  9   0  91   4   1   1   0   5   1   0]
 [  1   0   3  96   0   0   0   0   3   0]
 [  1   3   0   0  95   1   0   0   0   0]
 [  1   2   1   0   2  95   0   0   1  14]
 [  1   1   0   4   0   2  76   2  14   1]
 [  1   1   3   0   0   2   2 100   0   5]
 [  0   0   1   2   0   0   4   0  96   0]
 [  1   0   0   0   1   7   0   1   0  83]]

EDB.csv Nerr= 126

EDB.csv Ntotal= 1050

EDB.csv PEG= 0.1200
```

Figure 21: PEG value and confusion matrix after quantizing by six bits

Error rate==31.29%

3.2.8 quantize by 7 bits



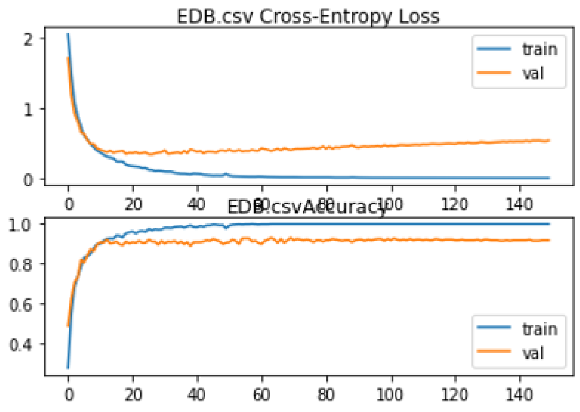


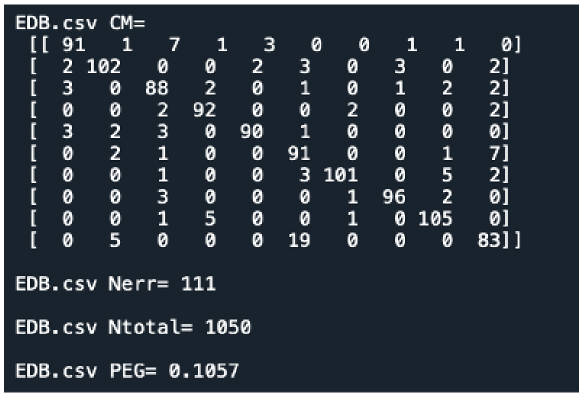
```
EDB.csv CM=
 [[104   0   6   0   0   0   0   4   0   1]
 [  0  96   1   0   2  10   0   1   0   2]
 [  1   1  85   1   1   0   0   1   1   0]
 [  1   0   2  85   0   2   1   0   0   0]
 [  4   3   0   0 105   0   0   0   1   0]
 [  0   3   1   0   0  87   0   2   0   7]
 [  1   1   0   4   0   1  98   1   9   2]
 [  2   0   0   0   0   0   1  84   1   1]
 [  0   0   2   6   0   0   2   0  96   0]
 [  1   2   0   0   0  17   0   1   0  95]]

EDB.csv Nerr= 115

EDB.csv Ntotal= 1050

EDB.csv PEG= 0.1095
```

Figure 22: PEG value and confusion matrix after quantizing by seven bits

Error rate=19.80%

3.2.9 quantize by 8 bits(1 bite)





```
EDB.csv CM=
 [[ 91   1   7   1   3   0   0   1   1   0]
 [  2 102   0   0   2   3   0   3   0   2]
 [  3   0  88   2   0   1   0   1   2   2]
 [  0   0   2  92   0   0   2   0   0   2]
 [  3   2   3   0  90   1   0   0   0   0]
 [  0   2   1   0   0  91   0   0   1   7]
 [  0   0   1   0   0   3 101   0   5   2]
 [  0   0   3   0   0   0   1  96   2   0]
 [  0   0   1   5   0   0   1   0 105   0]
 [  0   5   0   0   0  19   0   0   0  83]]

EDB.csv Nerr= 111

EDB.csv Ntotal= 1050

EDB.csv PEG= 0.1057
```

Figure 23: PEG value and confusion matrix after quantizing by eight bits

Error rate=15.64%

3.2.10 PEG-bit figure

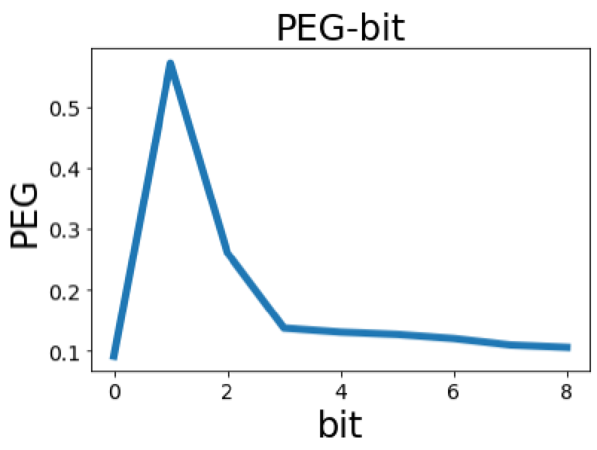


Figure 24: PEG-bit

Observing the PEG variation shows how much the spectrogram and the model can store and the effect on accuracy and speed. The figure also reveals that when n was not large, PEG value was much higher than I had expected, but when n was greater than 7, equivalent to setting more than 128 thresholds, PEG values gradually decreased to less than 0.1, approximately equal to the PEG value trained with the previous normalized spectrogram. Zero bit means that I did not quantize the data set and the training set is the original input data.

4 Conclusion

This project is mainly for improving the accuracy and efficiency of training. We use three methods to explore it: committee voting, more restart numbers, and Relu and SoftMax as activation functions.

For the first, our team committee voted to do speech classification. We look into what impact would be made by a different number of models voting during the process. We train 39 models with relatively high accuracy. Their PE values are lower than 0.083. With these models, we. Compare different PE values performed by the different number of models voting. The PE curves show a minimum PE value, and its corresponding number of models exists in the interval [20, 35]. General performance would not constantly go up but even down when there are too many models voting.

In addition, the more restart number we have, the more accurate the trained expert. Firstly, we constructed a simple version。 restart algorithm to output the graph. Then, the research on several differences is analyzed and compared with our figure. These two are coincident with each other, which testified my hypothesis. And perfect epoch number is selected using iteration through 100 epochs and seeing which one can give the highest test set prediction accuracy. Mode of 20 trials suggested 13 as epoch number. Overall, this research supports the existing essay on restart number and its indirect relation with expert accuracy in finding out global optima. The study shows that the accuracy can be improved with those various restart algorithms.

The last, use Relu and SoftMax as activation functions. After preprocessing and normalization of recording data, we delivered the best performance neural network training (including dropout) with a PEG of

0.0914. According to the change of PEG with a bit, after random quantization in bits, the PEG value fluctuated significantly. However, after the number of bits was more significant than 7, PEG fluctuation was stable within 20% (PEG=0.1095, 0.1057), and the training time was significantly reduced (the time of 100 restarts was shortened by nearly one-third), which makes the training results acceptable. This is also the storage limitation of the model in the spectrogram in the neural network.

In conclusion, the study results are consistent with the theory and provide a theory base for later improvement of the accuracy and efficiency of training. The original data in other models, such as image recognition, can be processed and compared, and the dimension of the data set can be reduced by pooling to save the time of operation and optimization and improve accuracy. I will also continue to study and try to solve these problems.

Reference

[1] Brownlee, J. (2019) Better deep learning. E-Publishing Inc., Texas. pp. 1–86, 141-166, 369-378

[2] 衣世东. 2018(1)基于深度学习的图像识别算法研究[J] 网络安全技术与应用, 1: 39-41

[3] Ian Goodfellow, J., Bengio, Y., Courville, A. (2016) Deep learning. Publishers of Electronics Industry, New York

[4] Brownlee, J. (2020) Deep learning with python. MIT Publishing. Texas.

[5]Dick, T., Wong, E., Dann, C. (CMU), 2014. How many random restarts are enough?.

URL: https://www.cs.cmu.edu/~epxing/Class/10715-14f/project-reports/DannDickWong.pdf

[6]Wagenmakers, E.J. (U of Amsterdam) ,2007. Stopping Rules and Their Irrelevance for Bayesian Inference: Online Appendix to “A Practical Solution to the Pervasive Problems of p–Values”, to appear in Psychonomic Bulletin & Review. URL: https://ejwagenmakers.com/2007/StoppingRuleAppendix.pdf