Machine Learning for Spoken Digit Classification

Adviser: Professor Roman Kuc[5]

Abstract

Our research aims to improve the accuracy of spoken digit classification and adopt many efficient methods. For training data, using data augmentation to expend the training size, including regularization augmentation and noise augmentation. For structure of neural network, our group compare three kinds of activation function including sigmoid, logistic and Softmax. In our training process, we compared Batch Gradient Descent, Stochastic Gradient Descent and Batch Stochastic Gradient Descent. As for the single neural network, the Softmax neural network with Batch Stochastic Gradient Descent gave the best performance and improved the baseline performance by 27%. In addition, according the limitation of single neural network, our group gives the voting method and the average method to cooperate the neural network with different local minimum. This cooperation method improves the neural network's generalization classification performance by 4 7%. Real-time classification, the best performance was consistently obtained for voting method with BSGD using Softmax.

Keywords: data augmentation; softmax; logistic; batch stochastic gradient descent; voting method.

[1]M

[5]Yale University

1 Introduction

Machining learning is becoming an essential component in the modern and future intelligent system and society, including speech recognition, computer vision, and natural language processing, these are relating to the neural network.

Machine learning about speech recognition in this paper is based on "Train Data Processing", "Structure of Neural Network" and "Training Method and Cooperation". The neural network we use have two hidden layers, and each hidden layer has 16 nodes. The input of the neural network has 100 nodes, and the output has 10 nodes. The problem is when we use this database to train a neural network. The database might be not enough to cover all kinds of features. For audio, features like different background noise, different frequencies and different accents, etc., may not be included entirely in some databases which cause overfitting the training data with poor generalization ability [1]. As well as collecting data is often costly. In addition to, we used Sigmoid function as activation function shows an issue that when the value of z is too high or too low, the derivative of y is close to zero which causes the gradient of w close to zero. This phenomenon is also called gradient disappearance [1]. Thus, gradient disappearance will slow the gradient updating. In multi-layer networks, the problem of gradient disappearance will be more serious. During the deep learning optimization, so far, the most successful approach for deep learning optimization is to use gradient descent. However, this is a greedy algorithm and hits some of the biggest open problems in the neural network, which means when we use gradient descent to train our deep neural network, it is not guaranteed that a better solution cannot be found. Moreover, the neural network may introduce the most well-known problem, the saddle point.

This research aims to get as small as possible PEG value using the limited database. In this work, in train data processing part, we suggest that increasing the number of training samples by going through the data augmentation techniques which include noise and regularization augmentation, obtaining an improvement on generalization ability. Besides, to develop our neural network by activating other functions rather than Sigmoid function, we are going to use additional functions not only can overcome or weaken gradient disappearance but also can speed up the neutral network or improve convergence ability. At the same time, with deep learning neural network, we are going to train three different gradient descent methods that will help us to recognize which one can effectively to get good enough local minimum and they just ignore the hardness results and run as many iterations of gradient descent to get the best local minimum neural network, the smallest PEG value. After that, we use the voting method to avoid the issue of a single neural network would only fit a part of data, and it would have different performance on different person's data.

To investigate these issues, we have performed a comprehensive set of experiments for changing to different kinds of training samples, neural network structures and training methods. Also did some comparisons between these different types that we have. The results provide evidence that demonstrates PEG value does promotions that only fewer errors are existing eventually.

The paper is organized as follows. Chapters 3-5 describe the main progression of the research for "Train Data Processing", "Structure of Neural Network" and "Training Method and Cooperation". Each chapter also includes comparison results and their trends. Chapter 4 presents the best experimental results on generalization performance for updating training samples, using neural network structures and gradient descent training methods in addition to the real-time classification. We discuss some significant errors found during our experiments. In general, the PEG value was reduced by about 47% compared to the baseline system and real-time classification performs even better, especially using the voting method. Finally, conclusions are drawn in Chapter 6.

2 Train data Processing

2.1 Data augmentation

When using data base to train a system like an NN, we hope that the larger the data size and the higher the quality, the better the generalized ability of the trained system. However, when actually collecting data, it is often difficult to cover all the scenes. For example, for noise conditions, when collecting audio data, it is difficult to control the type and size of noise; Also, for human voice, if you want to record It is very impractical to collect a lot of people's sounds of different frequencies, which requires a lot of cost.

Thus, overfitting can happen during training. Overfitting is due to the fact that the training data contains errors during the training of the NN using the data set, and the system also fits the errors during training. The main reasons are: The system itself is too complicated to fit the noise in the training data set; and The training samples are too few or lack representativeness. In this case, you need to increase the number of samples, or increase the diversity of samples. If we can use a certain amount of existing data to automatically generate a very large and wide variety of data, we can not only save the cost of obtaining data (such as recording, etc.), but also enhance the type of data to train a system to avoid overfitting. For the problem of lacking of data to train an NN, data augmentation is an effective method to expand the data sample size, and obtain better generalization ability.

2.2 Approaches

Two approaches to do audio data augmentation are described.

2.2.1 Noise augmentation

For noise augmentation, we can simply add some kind of noise to the existing audio data. Since we used ASTFT features to process the audio, noise values can be added to the ASTFT features.

To do this, I add the random noise in the process of extracting the existing audio data and transforming them into ASTFT features to train an NN. I first examine the length of every piece of audio signal, then add a random value of the same size to it.

To make sure that the noise being added is not too large which can distort the original values, we can observe the ASTFT values in advance and adjust the noise value into a proper one.

2.2.2 Regularization augmentation

Since the pitch is representing a piece of audio's feature, so what I want to do is to do something to the ASTFT features. In this case, regularization can be used here to improve the generalization of the NN. Regularization should be used to reduce the complexity of the NN, limit the parameters to too many or too large, and avoid the NN becoming more complicated. When using ASTFT features as the basis for training an NN, regularization can be used to limit certain ASTFT features.

The regularization I used here is to limit the ASTFT feature values. First, Average all the ASTFT feature vectors to produce AvFT. Then, Compute InvFT = 1/AvFT. In this case, since some of the ASTFT features "stand out", then their values can go much beyond the average, which causes that some InvFT will be very large. To limit those relatively large values, produce the regularized value RInvFt:

$$RInvFt = min(\tau, InvFT) \tag{1}$$

where τ equals four times the min value in InvFT. In this step, all the values all limited under τ. Then, multiply the original ASTFT by RInvFt to produce regularized ASTFT feature values used to form a data base.

3 Structure of Neural Network

3.1 Tanh function

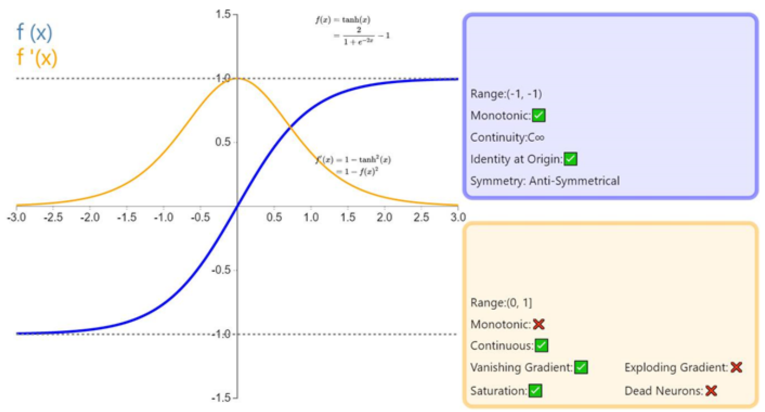


Figure 1: tanh function

Tanh function shown in Fig. 1 is a downward translation and contraction of sigmoid function which can be regarded as linear in a small interval around 0. Since the mean value of tanh function is 0, it doesn't have the shortcoming of sigmoid function whose mean value is 0.5. However, it still have some disadvantages of sigmoid. That is, when z is too large or too small, the gradient is almost zero, so it is slow to update the network using the gradient descent optimization algorithm.

$$tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \tag{2}$$

The convergence speed is faster than sigmoid function for two reasons.

1. The gradient disappearance problem of tanh (z) is less serious than sigmoid. If the gradient disappears occurs early, the convergence rate is slower.

$$tanh : tanh'(z) = 1 - tanh(z)^2 \in (0, 1) \tag{3}$$

$$sigmoid : s'(z) = s(z) * (1 - s(z)) \in (0, 1/4) \tag{4}$$

2. The influence of zero mean. If the best optimization direction of the current parameters (w0, w1) is (+ d0, -d1), according to the back propagation calculation formula, we want x0 and x1 to have opposite signs. However, if the upper-level neuron uses the sigmoid function as the activation function, and the sigmoid is not centered on 0, and the output value is always positive, then we cannot perform the fastest parameter update, and instead approach the optimal solution by Z-shaped.

3.2 Logistic function

Figure 2 shows the logistic function. Logistic function investigated is just like sigmoid function apart from one difference - the coefficient α, as

$$y = \frac{1}{1 + e^{-\alpha * z}} \tag{5}$$

As the function above, α is a coefficient which is less than 1. By changing the value of α, the function saturates less quickly. This feature will weaken gradient disappearance. Thus, its convergence ability is better. One thing important is to find the value of α that gives best convergence. We should also notice if the best value of eta (learning rate) will change at the same time. However, it still have some disadvantages as sigmoid. Although it weakens gradient

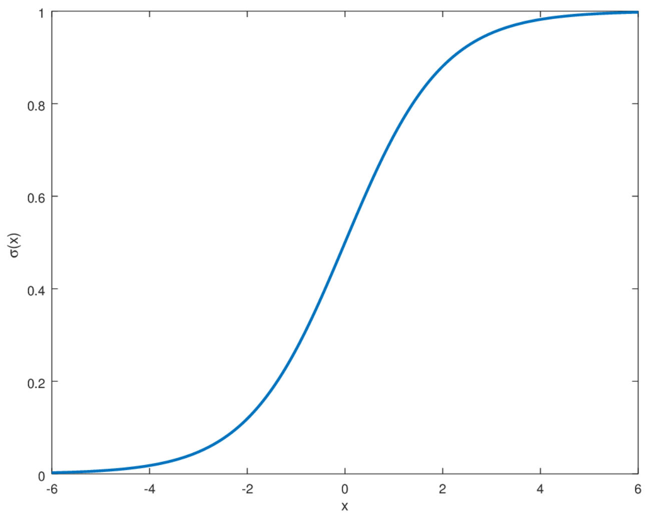


Figure 2: Logistic function.

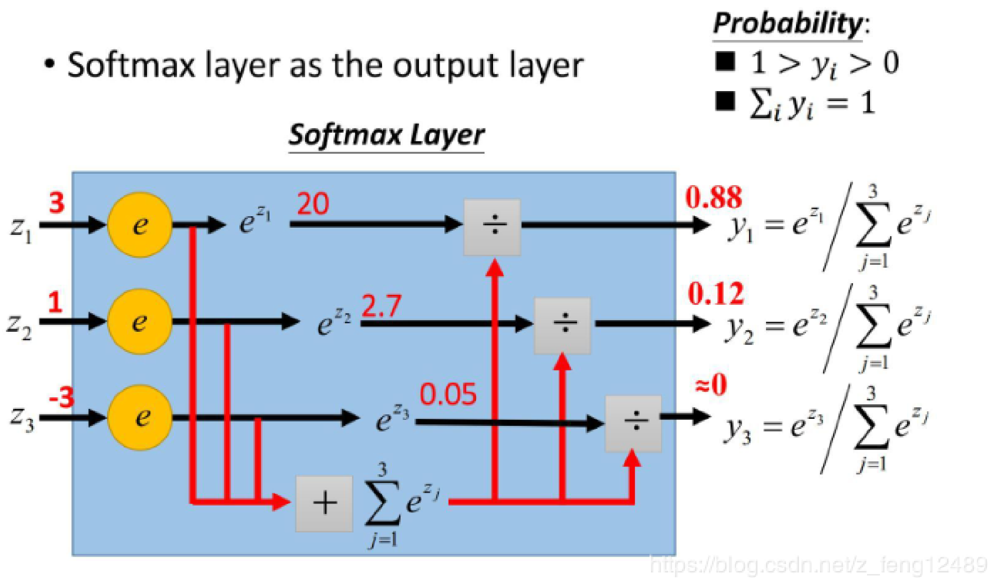
disappearance by stretching left and right, the up and down position of the function has not changed. Thus it is still a non-zero mean function.

3.3 Softmax Method and Cross-Entropy Function

With multiple classification, the sigmoid function measures the different among the data. Hence, the function determines the accuracy of neural network on some content. Our group applied the Softmax function and use cross-entropy function to measure the cost of neural network. This function can represent more subtle differences than sigmoid function by doing some calculation. In other words, this means that the neural network can define more accuracy classification. For example, it can distinguish 6 and 8 if the nonlinear function can present the subtle differences between them.
The Softmax function uses exponential functions to calculate the probability of each output. In the Softmax, the k is an index over all classes and the sum of the output equals to 1.0 as

$$p_k = \frac{\exp(z_k)}{\sum_i \exp(z_i)} \tag{6}$$

The output of the neural network is probability, to measure the discrepancy between the target data and the actual data, our group apply cross-entropy function to calculate the loss.



• Softmax layer as the output layer
Probability:
■ $1 > y_i > 0$
■ $\sum_i y_i = 1$
Softmax Layer
z_1 3
z_2 1
z_3 -3
e
e^{z_1} 20
e^{z_2} 2.7
e^{z_3} 0.05
$\div$
$+$ $\sum_{j=1}^{3} e^{z_j}$
0.88 $y_1 = e^{z_1} / \sum_{j=1}^{3} e^{z_j}$
0.12 $y_2 = e^{z_2} / \sum_{j=1}^{3} e^{z_j}$
≈0 $y_3 = e^{z_3} / \sum_{j=1}^{3} e^{z_j}$
https://blog.csdn.net/z_feng12489

Figure 3: The schematic of softmax method

$$Loss = -\sum_j d_j \log p_j \tag{7}$$

where d_j is the target probability and the p_j is the output of the Softmax function. Expanding the gradient through the chain rule. We can obtain

$$\frac{\partial Loss}{\partial w_j} = \frac{\partial Loss}{\partial p_i} * \frac{\partial p_i}{\partial x_j} * \frac{\partial x_j}{\partial w_j} \tag{8}$$

We just first calculate the first two items, and the equations become the following:

$$\frac{\partial Loss}{\partial z_k} = \sum \frac{\partial Loss}{\partial p_i} * \frac{\partial p_i}{\partial z_k} \tag{9}$$

In this last function expanding the equation through calculating the derivative of Softmax function and cross-entropy function gives:

$$\frac{\partial Loss}{\partial z_k} = \sum_i \frac{t_i}{p_i} * p_i p_k (i \neq k) + \frac{t_k}{p_k} p_k * (p_k - 1) = p_k - t_k \tag{10}$$

It gives: the gradient descent is $p_k - t_k$, the p_k is the output of neural network and the t_k is the target probability of the number.

The neural network we use has two hidden layers and each hidden layer have 16 fully connected nodes. The input of the neural network has 100 nodes, the output has 10 nodes. In the hidden layer, we use sigmoid function and in the output layer, after linear combination, these vector are put in Softmax function. The output of Softmax function, the probability is the final output of the neural network.

4 Training Method and Cooperation

4.1 Training Method Theorems

We used the gradient descent method to train the adopted algorithm. Due to the small amount of training database, we used three different forms of gradient descent to compare which training method can approach to optimal value as the closest.

4.2 Baseline Algorithm

According to the assumption function of the general linear regression function is as follows:

$$h_\theta = \sum_{j=0}^{n} \theta_j x_j$$

The loss function corresponding to gradient descent is:

$$J_{train}(\theta) = 1/(2m) \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

4.2.1 Batch Gradient Descent

Our goal is to make the loss function as small as possible, that is, to solve the weights to make the loss function smaller. First, we randomly initialize the weights, and then update the weights repeatedly to reduce the loss function and until when it reaches the requirements. For the update algorithm here, we choose the gradient descent algorithm, use the initial weights and update the weights repeatedly: Here, α represents the learning rate, and represents the size

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

of the step in the steepest direction each time. In order to update the weights, we need to find the partial derivative of the function. First of all, when we only have one data point (x, y), the partial derivative of J is:
Then for all the data, the partial derivative of the above loss function is summed as:
The whole process of updating the weight is as follows:

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} \tag{11}$$

For random initialization of ANN weights, we chose two methods and compared them. In our test, one is Xavier Initialization (x = sqrt1/n*randn(k,n))[2] and the other is He Initialization (x = sqrt2/n*randn(k,n)) [3].

4.2.2 Stochastic Gradient Descent

Since batch gradient descent uses all the number of samples every time the parameters are updated, the training speed will become slower as the number of samples increases. Stochastic gradient descent is proposed to solve this method.

$$\begin{aligned}
\frac{\partial}{\partial\theta_j}J(\theta) &= \frac{\partial}{\partial\theta_j}\frac{1}{2}\left(h_\theta(x)-y\right)^2 \\
&= 2\cdot\frac{1}{2}\left(h_\theta(x)-y\right)\cdot\frac{\partial}{\partial\theta_j}\left(h_\theta(x)-y\right) \\
&= \left(h_\theta(x)-y\right)\cdot\frac{\partial}{\partial\theta_j}\left(\sum_{i=0}^{n}\theta_i x_i - y\right) \\
&= \left(h_\theta(x)-y\right)x_j
\end{aligned}$$

$$\frac{\partial J(\theta)}{\partial\theta_j} = -\frac{1}{m}\sum_{i=1}^{m}(y^i - h_\theta(x^i))x_j^i$$

It uses the loss function of each sample to obtain the corresponding gradient of the partial derivative of θ to update θ:

$$\theta_j^{'} = \theta_j + (y^i - h_\theta(x^i))x_j^i$$

So that the whole process of updating the weight is as follows:

$$\theta_j := \theta_j - \alpha(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)} \tag{12}$$

4.2.3 Batch Stochastic Gradient Descent (Mini-batch Gradient Descent)

BSGD has combined the advantages from both BGD and SGD, has thetrainning process of the algorithm is faster, and the function of ensuring the accuracy of the final parameter. Whenwe update the parameters here, we did not take into account all the training samples and then summed divided bythe total number. BSGD is randomly selecting data from overall database then using batch sizes samples to update.
We just randomly select data from the whole data and use decreasing learning rate to move smoothly. And change the weights every 100 train data by averaging the changes. This can help the train process have low oscillation and change the weights more efficiently.

4.3 Comparing BGD, SGD and BSGD

The comparison chart of the three methods is shown in Fig. 5.Batch gradient descent is all of our training data is used for each parameter update. One disadvantage of this method is that it is very time-consuming. Stochastic gradient descent is iteratively updated once by each sample, so there is a disadvantage that one iteration may not accurately achieve the optimal, so stochastic gradient descent is not directed toward the overall optimization direction for each iteration. Batch Stochastic gradient descent is faster than BGD and easier to approach the optimal than SGD.

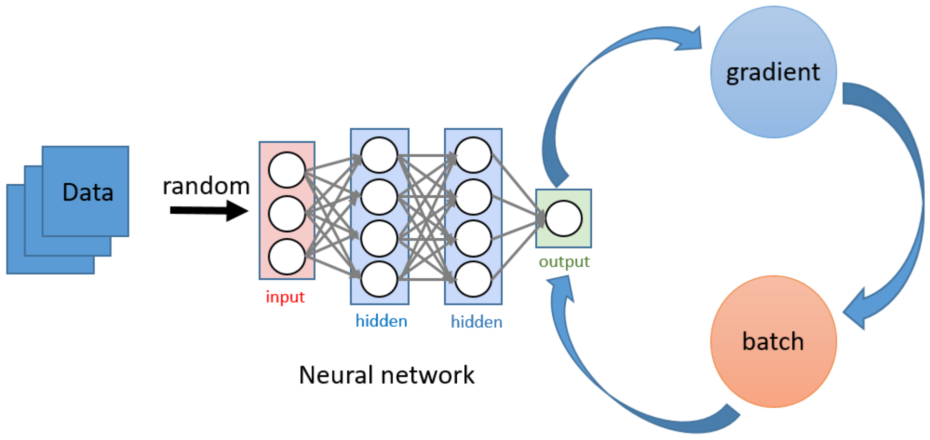


Figure 4: The flow chart of BSGD.

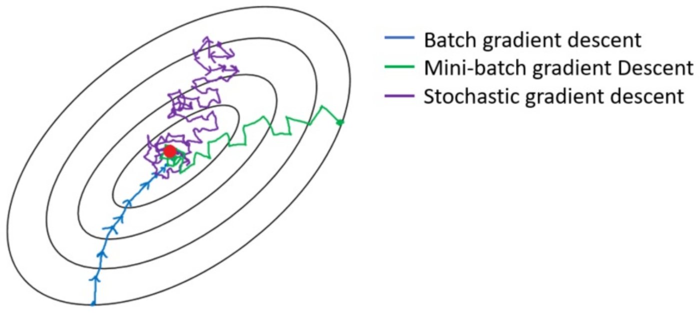


Figure 5: SGD, BGD and BSGD comparison.

4.4 Performance with Different Training Methods

4.4.1 Using Sigmoid function training with He initialization and Xavier initialization in BGD

Figrue 6 shows only one time that the PEG value of Xavier initialization is lower than He, other than that is it greater than He. So He initialisation performs better than Xavier initialisation in the deep learning neural network.

4.4.2 Using Softmax Method and Cross-entropy function with BGD

Figrue 7 shows that BGD helps Softmax function and Cross-entropy function to get out off saddle points to produce more stable PEG results. Softmax function does help to help us get closer to optimal value.

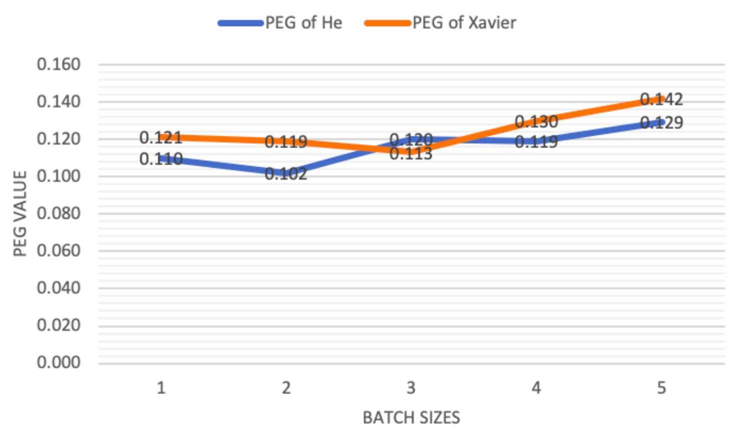


Figure 6: He initialisation and Xavier initialization performance in BGD.

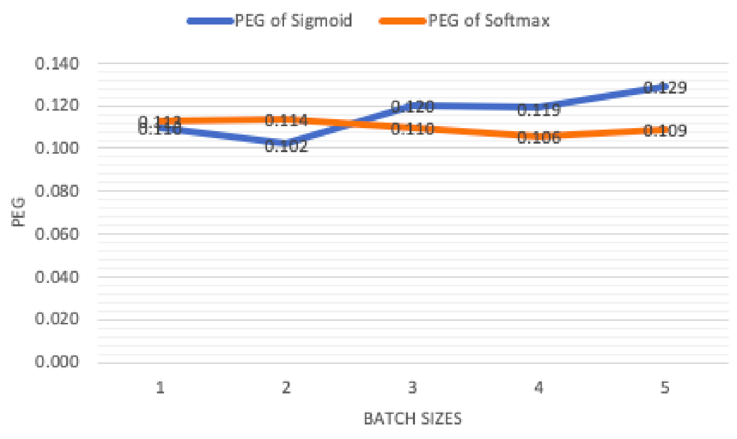


Figure 7: Sigmoid function and Softmax function performance in BGD.

4.4.3 Using Softmax Method and Cross-entropy function with SGD and BSGD

From Figure 8, BSGD has achieved the lowest PEG value, PEG=0.104, which means BSGD is the best method to optimise the parameter by using our data.

4.5 Cooperation Method of Neural Networks

Based on the approach above, Many engineers have given effective method to get good enough local minimum, the smallest PEG value. However, the single neural network would only fit a part of data and it would have different performance on different person's data. Thus, using one single neural network would not have enough ability to fit all the features of data and have the instinct problems. Our group introduce the cooperation method of neural networks.

Our group put foreword 2 kinds of cooperation methods:
(1) Voting method.

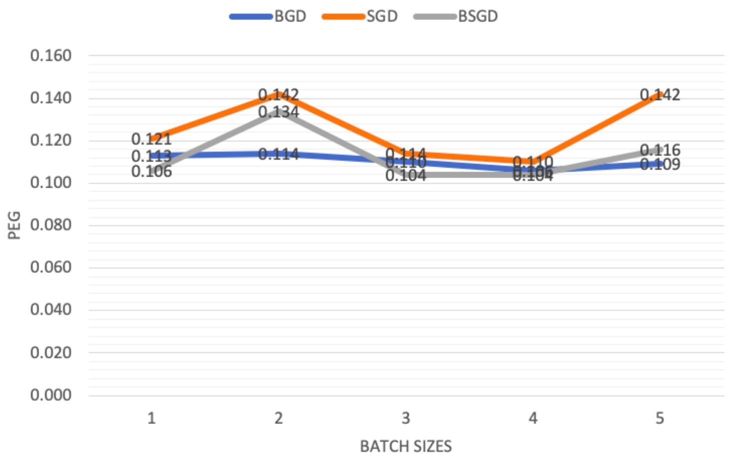


Figure 8: Softmax function performance in BGD, SGD and BSGD

(2) Average method, average the output of each neural network.

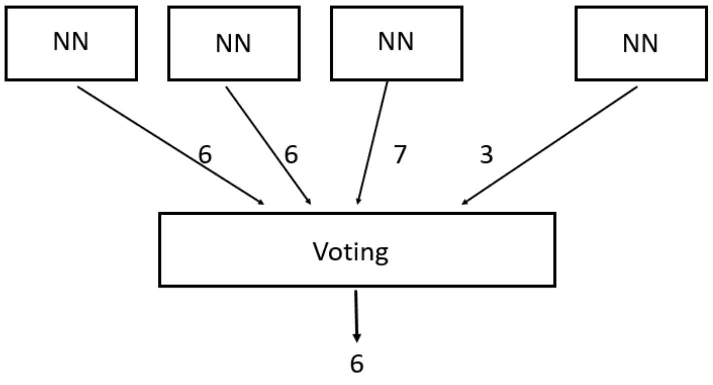


Figure 9: Example for voting with several neural networks

4.5.1 Votign Method

The voting method uses the output of each single neural network to do voting as shown in Fig. 9, if the digit 6 have the most of the votes, the cooperation of neural networks' output is 6. The neural network confines the learning rate is 0.1 and the number of the epochs is 2000. It has two hidden layers and each hidden layer have 16 nodes. The input of the neural network has 100 nodes, the output has 10 nodes.

4.5.2 Number of Neural Network in Cooperation

Figure 10 shows using average method, increasing the number of neural network in cooperation, we have the PET and PEG values

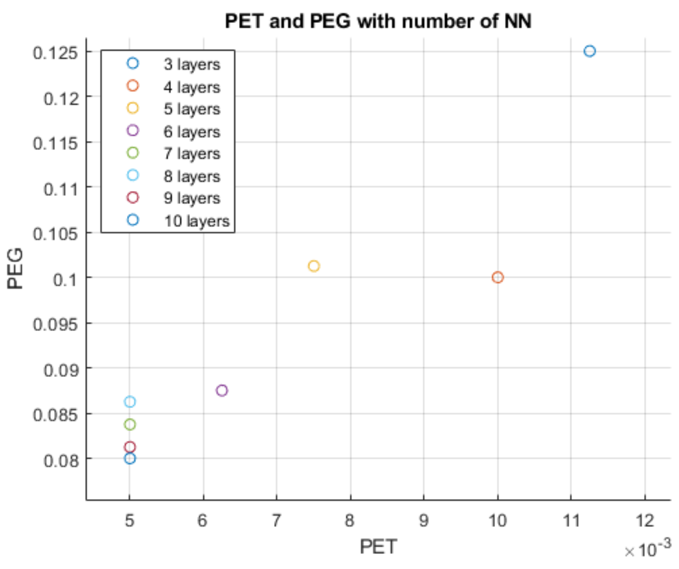


Figure 10: Average method with several neural networks.

Figure 11 shows the PEG value is decreases as the number of neural network increases. Also, with number increasing, the speed of PEG is decreasing. Also, test the voting method with the increasing number of neural networks. From Figure 11 we can find that with the number of neural network increasing, the PEG value is decreasing

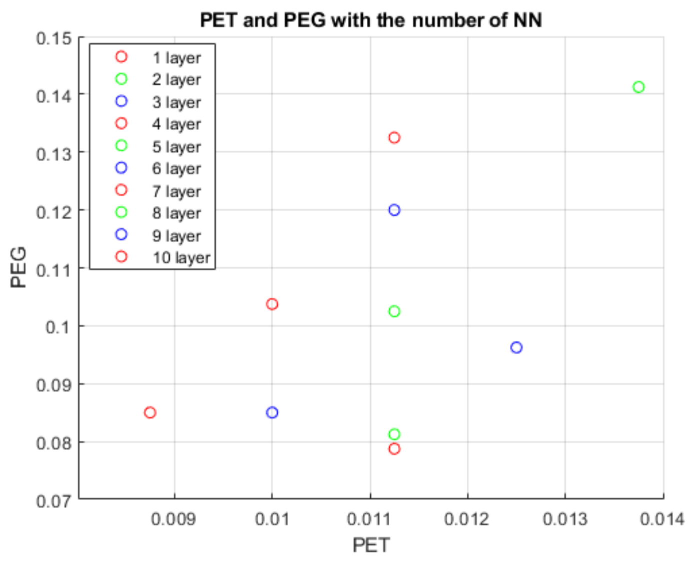


Figure 11: The changes of PEG with number of NN

using voting method. Also, the same as the average method, with number increasing, the decrease speed of PEG is decreasing. Consider the computational simplicity, our group adopt voting method as the cooperation method.

5 Results

5.1 Results of Data Augmentation

For noise augmentation, we add the random noise in the process of extracting the existing audio data and transforming them into ASTFT features to train an NN.

1. First, regularization improved the generalization of the NN also in the process of extracting the existing audio data and transforming them into ASTFT features to train an ANN.

2. Then, the length of every piece of audio signal was examined to generate random noise signals of equal length. The two signals are added to from a new signal with noise.

3. Noticing that the ASTFT feature values in the data base are in the order of 0.01, the noise value was adjusted to avoid additional distortion to the data base.

4. To see the results of these two approaches, the original data base was used as the testing data base, and take the noise-augmented data base and regularization-augmented data base as the training data bases(experimental groups). Also, there is original data base taken as a training data base as a control group.

Testing the original ANN gave the following results:

```
Epoch= 3000  PET= 0.014    PEG= 0.121 numRep= 3000   eta= 0.10
```

Test the NN trained by regularization-augmented data base: the detailed statistics are like this:

```
Epoch= 3000  PET= 0.016    PEG= 0.100 numRep= 3000   eta= 0.10
```

Test the NN trained by noise-augmented data base: the detailed statistics are like this:

```
Epoch= 3000  PET= 0.015    PEG= 0.107 numRep= 3000   eta= 0.10
```

As we can see from the results, after augmenting the training data bases by using regularization and noise, the PEGs decrease, which means that the generalization ability of the NN is improved. To reach a better generalization ability, combining these two approach to augment can be implemented.

First, noise-augmentation was applied directly after doing regularization-augmentation. The result is like this:

```
Epoch= 3000  PET= 0.020    PEG= 0.149 numRep= 3000   eta= 0.10
```

The PEG value is greater, which indicates a worse generalization ability. Maybe when the two approaches are implemented to a data base at the same time it causes distortion beyond ”augmentation”.

Then, a noise matrix of the same size with the ASTFT feature matrix, was added followed by regularization. The result is the following:

```
Epoch= 3000  PET= 0.020    PEG= 0.130 numRep= 3000   eta= 0.10
```

The PEG tells us that this combination may also bring in distortion that beyond ”augmentation” to the training data base. Thus, in the augmentation process, just use one approach to an NN at one time.

5.2 Testing

To test a noise-augmented NN's generalization ability, the noise augmented test data base was used to test it. We tested the original (none-augmented) NN, the NN trained by the data base augmented by 0.01 times the noise, and also the NN trained by the data base augmented by regularization. The comparison is shown Figure 12, the y axis shows the PEG while the x axis shows different test DBs (added by different noise multiple). As we can see, when

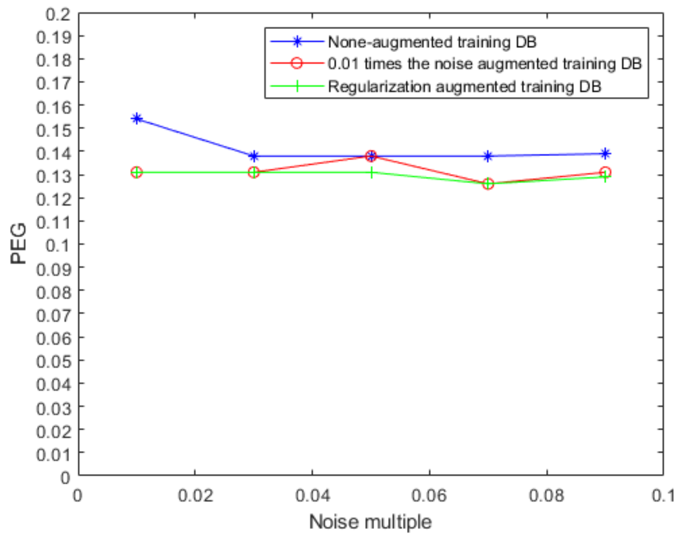


Figure 12: Test comparison

augmentation is applied to the training data base, the generalization ability towards different test DBs becomes better.

5.3 Finding activation function with best performance

Tanh function is the function with the *fastest* convergence speed in all 4 functions above, and it is a zero mean function which has better convergence ability than sigmoid. Logistic function is the function with *best convergence ability* in all 4 functions above. Although its speed is a little slower than tanh, in applications that don't care much about speed it is the better choice. We will search the verify and test processes below to observe their performance.

5.3.1 Tanh function

When we use sigmoid function and eta equals to 0.1, PET equals to 0.024. Compared with sigmoid function, tanh has better performance when eta equals to 0.08. And actually after experiments we find that eta = 0.1 is the best choice for sigmoid and eta = 0.08 for tanh function. When eta equals to 0.08 PEG of sigmoid function is about 0.180. While as we can see from the matrix below the value of PEG is 0.163 when we using tanh function.Thus tanh function does better in convergence.

Figure 13 shows some typical experiments using tanh function. The results show an issue. When we use sigmoid function we tried eta from 0.01 to 1.5 and the neural network is alive, but when we use tanh function if the eta is larger than 0.3 or less than 0.05 the neural network is dead. It shows that the limit of eta is stronger when we use tanh function.

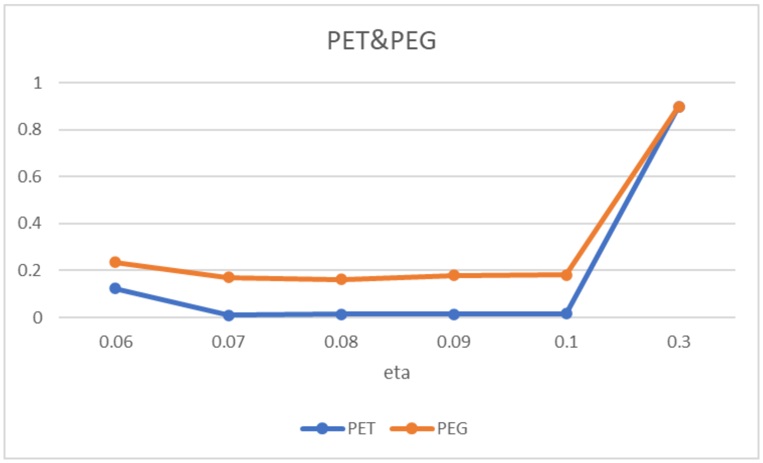


Figure 13: PET and PEG with different eta, NN2HL ASTFT tanh

5.3.2 Logistic function

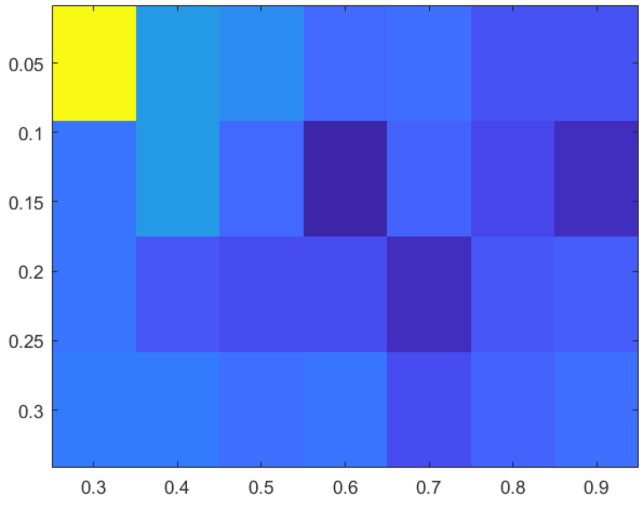


Figure 14: PEG as a function of eta (vertical axis) and alpha (horizontal axis).

As the research result shows in part 4, logistic function has better performance in convergence ability.But we should find what is the best value of alpha.I changed the value of eta and alpha at the same time to see what will happen. At each point(alpha,eta)I tried the experiments ten times. From the figure above, we can see that when eta=0.1,alpha = 0.6, the color is the deepest. It means that at this point we can get the best PEG = 0.0887. When we use sigmoid function (alpha=1)and eta = 0.1, we can get the best PEG = 0.18. Compare these two PEG, by using tanh PEG reduced by 50.73When alpha is less than 0.3, convergence speed and convergence ability slows down obviously.Thus alpha don't need to keep falling.

5.3.3 Combination of tanh and logistic function

Tanh function solved problem 'Non-zero mean' and logistic function solved problem 'gradient disappearance'. We consider if we can use alpha on tanh function as a combination of both of them. We tried, but the result of this function is not significant.

Results of tanh-logistic ,tanh, logistic:

alpha	eta	PET	PEG	
0.9	0.1	0.013	0.175	tanh-logistic
none	0.1	0.015	0.181	tanh
0.9	0.1	0.015	0.0938	logistic

As the results above show, the convergence ability of tanh-logistic is a little bit better than tanh, but worse than logistic. Besides, time waste of tanh-logistic and tanh is 50 second more than logistic and sigmoid (70 second). And when we tried to pan up and down (use another parameter beta), the neutral network will be dead. Thus, we think logistic is still the best choice.

5.4 The best performance after using BGD, SGD and BSGD as training methods

By putting PEG results of three methods, we found that BSGD is the best way to reach the optimal value. The train process of BSGD is below:

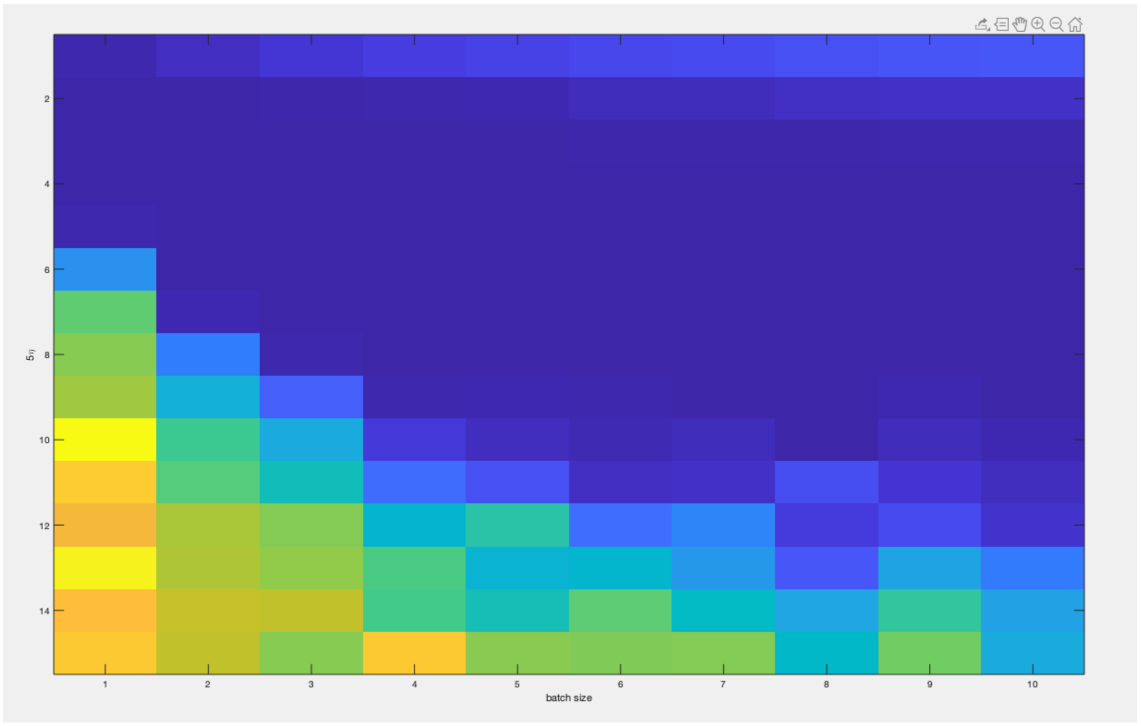


Figure 15: Randomly select data average of 5 restarts that give the Avecost observed over one epoch (eta/5=0-3, batch size=1-10)

In the BSGD, the cost of neural network varies, and we test the PEG of the test data, by testing different batch sizes with different eta values that selected from Figure 15. For example, we chose from the darkest square from column batch=3 that eta/5=1.4 as generating BSGD weights. The linear graph is shown in Fig. 17. By finding the

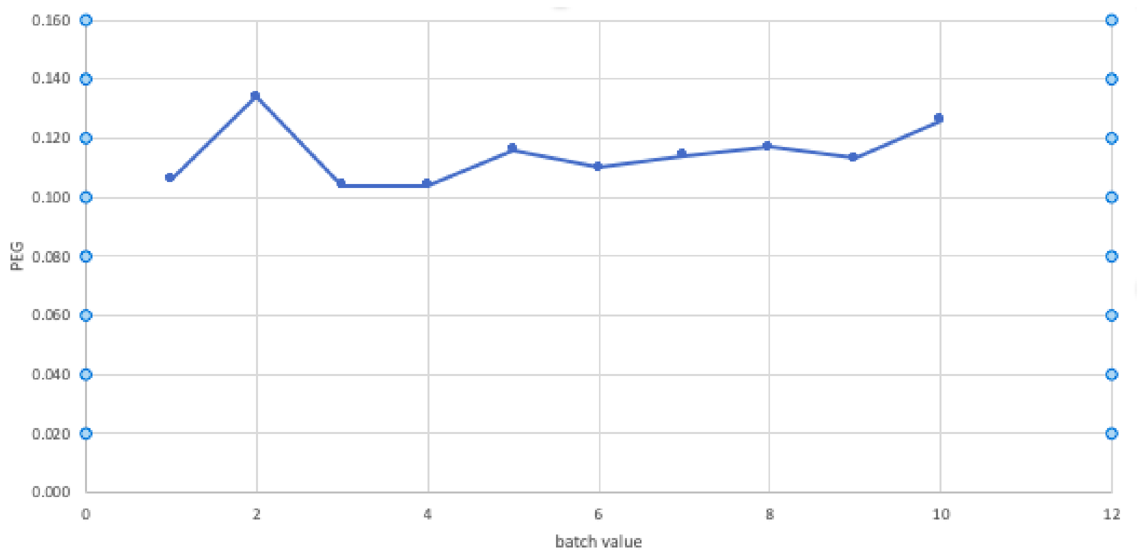


Figure 16: The PEG by different batch sizes

PEG value in weights of batch3 is the lowest, we finally got PEG=0.092, which is not including data from 7th data set , because we found significantly error that affects final result.

5.5 Cooperation of Networks with Different Structure and Training Method

Based on the training methods including gradient descent, batch gradient descent, stochastic gradient descent, and batch stochastic gradient descent and different function including Softmax function and sigmoid function. Also, with noise method, we calculate the PEG of different neural networks in cooperation. Figure 17 shows the results. The figure includes four kinds of neural network using the following methods.
1. Softmax neural network with gradient descent method.
2. Sigmoid neural network with gradient descent method and noise data.
3. Softmax neural network with batch stochastic gradient descent method.
4. Sigmoid neural network with gradient descent method.

By comparing the different neural network with different method, we can have such conclusion:
1. The PEG is decreasing with number of neural networks increasing
2. The PEG's decreasing speed is decreasing and around 10 neural networks, the PEG tends to be stable
3. The PEG of the single neural network using Softmax and batch stochastic gradient descent method has the smallest value.
4. The cooperation of these neural network perform nearly the same when the number of the neural network in cooperation arrival a large number. For this problem, the number is 10.
5. 10 neural networks will make the PEG saturation when there are small data set.

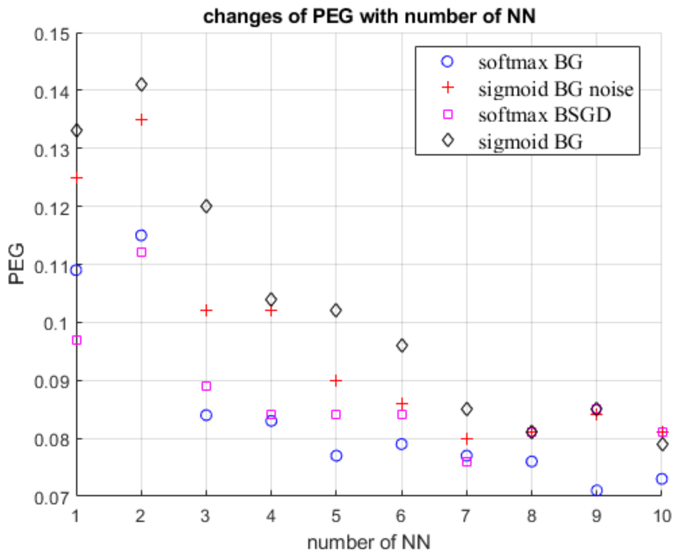


Figure 17: The PEG by different batch sizes

5.6 Real Time Classification Using a Single Neural Network and Voting Method

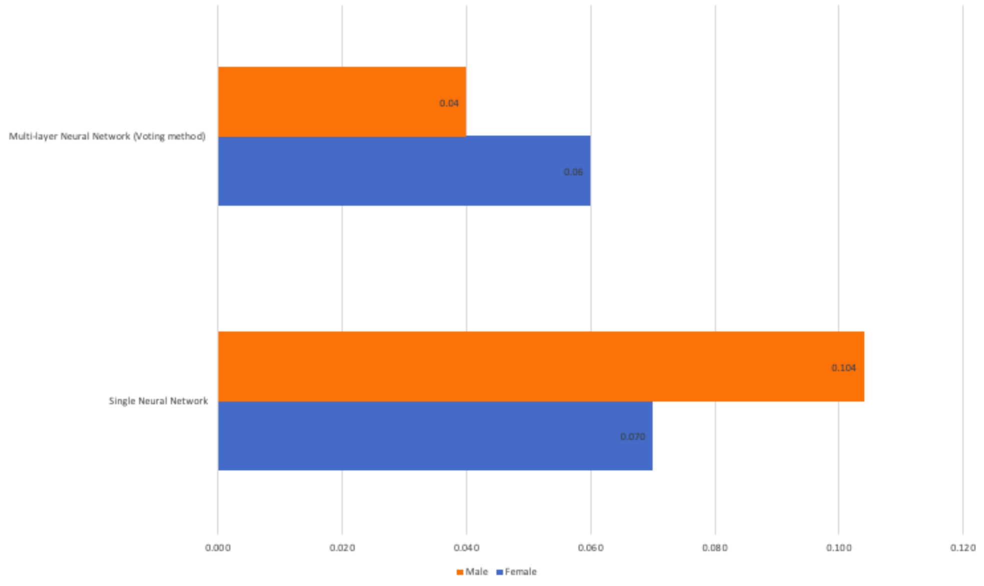


Figure 18: The PEG results from RT classification (trials=100)

By testing the real time classification using a single neural network and voting method, the results are shown in Fig. 18. Testing by using BSGD weights from when batch size is 3 and the number of neural network in cooperation is 10, the PEG values including female and male speech is shown. The best real time classification uses the voting method to produce PEG equals 0.04, which is lower than the calculation result. It is a pretty good result. However, from everyone's feedback we discovered by using a single neural network is difficult to avoid errors if you just randomly say 6 and 9. By using the voting method, when we test the digit 6, it is likely to be defined as 8. Although, the result is 6, the votes between 6 and 8 is almost the same, the output of neural network is below:

```
E_NN2HL_ASTFT classified Test digit 6 as 6
    CORRECT :-)

RT_Classification_E_NN2HL_ASTFT
 ANN# outputs
   0    1    2    3    4    5    6    7    8    9
 0.00 0.00 0.00 0.00 0.00 0.00 5.00 0.00 4.00 0.00
```

From the output of neural network, the votes of 6 is slightly larger than 8. The neural network in cooperation improves the performance of classification.

6 Conclusion

We have presented an empirical study of the performance of two hidden layers neural network and gives the practical methods to improve the accuracy of neural networks.

In this study, our group use data augmentation method including noise and regularization, different structure of neural networks with logistic and Softmax, and 3 kinds of training methods : batch gradient descent, stochastic gradient descent, and batch stochastic gradient descent. Among them, the softmax neural networks with batch stochastic gradient descent training method have the best performance, the PEG is 0.097 (the base PEG is 0.133, improving by 0.27).

In addition, with different kinds of neural networks, our group come up with cooperation of many neural networks including average method and voting method. By cooperating the neural networks, the performance of this neural networks have improved compared to baseline ANNs by 0.47 (PEG of 9 ANNs in cooperation is 0.071).

Overall, the research results support the broad conclusion and provides practical cooperation method to improve the performance. From the running time of cooperation method, the time is proportional to the number of the neural network in cooperation. Also, this can be accelerate by using parallel computing.

References

[1] L. Bottou. Stochastic gradient descent tricks. In G. Montavon, G.B., and K.R. Müller, editors, *Neural Networks: Tricks of the Trade. Lecture Notes in Computer Science*, volume 7700, pages 421–436. Springer, 2012.

[2] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. *AISTATS*, 9:249.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *ICCV*, pages 1026–1034, 2015.