Progressive Image Compress:

Generate and Simplify a Model under Evolutionary Algorithm and Gradient Descent

Ruikang Zhou
Research School of Computer Science
Australia National University
u6976157@anu.edu,au

Abstract. Progressive image Compress is a technique that can compress the data with different compression ratio by reducing the amount of neurons in hidden layers. It can decrease the size of the parameters of the model and improve its training efficiency. We design a model based on the anger data set and train the model to get a great performance which is about 80% accuracy.Then we implement a hidden layer compress method to decide which neuron in hidden layer can be removed by analyzing the similarity of hidden units. The result shows that the compression have little influence before we reach the simplest structures. We also use the evolutionary algorithm to train a great model with 80% accuracy and test the simplest structure we got for this task under before. And what we found shows that the progressive compression is unique to a task and cannot share the structure between different tasks even they look almost the same.

Keywords: .Progressive image compress, anger database, Pruning, Evolutionary Algorithm

1 Introduction

Anger is a common emotion of human beings and it always can be a signal to show how a man is feeling. By detecting a man is angry or not can help us to find a better way to communicate and work. However emotion can be fake and so is anger. People would prefer to cover their anger with gentle words and behaviors on purpose sometimes. Thus how to tell a man is really angry or not can be an inspiring work to do.

Micro-expression is one of the signal that can reveal a man's real emotion. And pupillary response patterns is an important micro-expression that might be ignore [1]. However there are many data can be collect from one's pupil, and is hard for us to analyze. Thus we are using an neural network based on an anger database to help to predict the anger is posed or genuine.

Except traditional way, we are using an evolutionary algorithm to train the model at the same time. Evolutionary algorithm uses mechanisms inspired by biological evolution. It is a simulation of natural gene reproduction.The possible solutions of question and possible parameters to be addressed are coded into string. Then we arrange a population size of the gene and simulate the gene mutation, gene exchange and inheritance [8]. By arranging the fitness of the gene we can start evolution and will find a good solution after hundreds generations.One of the advantages of evolutionary algorithm is that it is easy to realize and not restricted to particular optimization problem. And in this report, we are using the evolutionary algorithm as one of the training method and compare with the traditional gradient descent.

The use of neural network in computer version is promising and inspiring. However great performance of neural network can sometimes lead to tedious and huge calculate which means great pressure on server and great cost on time. Thus we are using hidden layer pruning to simplify the network structure and trying to reduce the number of calculating. After training the model with two different ways, we will use the progressive hidden layer compression to reduce the number of neurons of the hidden layer [3]. Then we will try to find the minimum size of the model that can solve the anger detect problem and to see if this kind of simplest model structure is suitable for evolutionary algorithm. By comparing with the results we can draw a conclusion whether and how the hidden layer pruning influence the performance of the model and can the model structure and compression strategy be used in another way of training.

2 Method

The way we implement a model that can detect the anger is genuine or posed and compare the influence of hidden layer pruning under two different training ways can be divided into 3 parts. The first part, we train a model with traditional gradient descent and test the performance of it. The second part, we retrain the model with evolutionary algorithm and record its accuracy. On the third part, we use the progressive image compression to reduce the hidden layer neurons and get the simplest model for this task(With the model we design before as the start).Comparing all the results to see the influence of the hidden layer pruning on the model and whether a simplest structure can be used in another task even though the input and output are same.

2.1 Detect the Genuine or Posed Anger

Data Description

The model is trained with anger database provided. According to the description, the database contains 400 items and each item has 9 columns and one label. The data is mainly about their pupillary response patterns which was recorded from 20 individuals watching 20 different short videos and 10 of them perform posed anger and the other 10 perform genuine anger. The videos are all from YouTube and selected manually based on identifiable and reputable sources. And the videos of genuine anger from documentaries and live news, while the posed anger videos are made from cut of movies [1]. Besides all the factors that may influence the final results have been removed as much as possible. 9 attributes of the items are showed below:

-O: O means the serial number of the observer
-Video: Video means which video one is watching
-Mean: The average value of the observer's pupil diameter
-Std: Standard deviation
-Diff1: Change of the left pupil diameter
-Diff2: Change of the right pupil diameter
-PCAd1: Orthogonal linear transformation with first principal component
-PCAd2: Orthogonal linear transformation with second principal component
-Label: What kind of anger the video really means.

First of all, O and Video are not usable and should be ignore for they have nothing to do with what we are going to find out. We compare the other 6 attributes with their distributions and can find that the difference between distribution of PCAd1 of posed anger and genuine anger is easy to find while the other attributes is almost the same.Which indicate that PCAd1 might be one of the most important index to detect the real anger. However we cannot just ignore the other attributes for the combine of calculate in high order might tell genuine and posed apart.

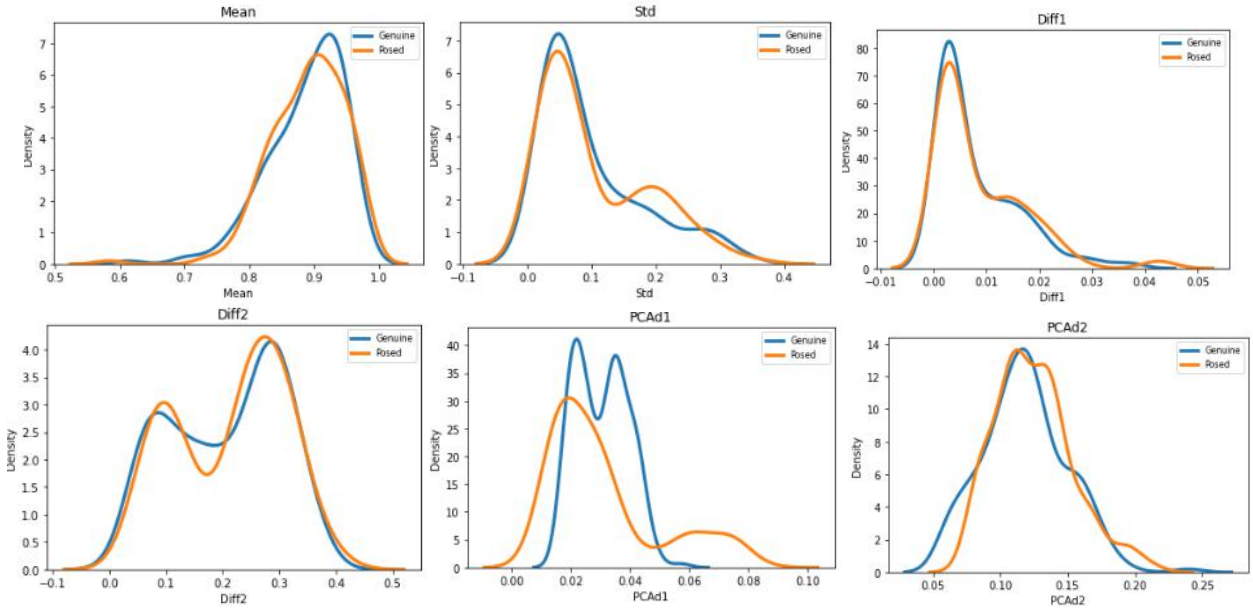

Figure.1. Distribution of the Attributes

Data Processing

```
mean: 0.8890901472513795std: 0.10246244160256897diff1: 0.008421385593939535
diff2: 0.20957463402082863pcad1: 0.030703412902810805pcad2: 0.12138183254195667
```

Figure.2 Average value of Attributes

Before we feed the data into the model for training, we have to do the data processing. The size of each attribute can be so different that might lead to over-fitting. Thus we have to decide whether we have to normalize the data so that the parameters in the model can be updated with almost the same step and will not result in many "die" neurons.From the figure.2 we can find that the mean of mean is much larger than diff1 thus we are using min-max normalization(Equation 1) to normalize the attributes to have the almost same range.

$$x_i = \frac{x_i - x_{mean}}{\sigma_x^2} \tag{1}$$

As for the label, we use the 1 and 0 to represent the posed and genuine anger. Number on each index means the probability that it equals to true of false. The way we setting like this is inspired by the form of output of the model that pass the sigmoid function. We will select 80 items randomly as test set and the other 320 items will be train set. Before the training we will shuffle the data set to reduce the possibility of over-fitting.(Continuous posed anger data or genuine anger).

Model Structure

The model contains 4 layers, one input layer, one output layer and two hidden layers. The number of three different layers are 6, 6 ,1. The number of neurons of first hidden layer is decided after several test and is complex enough that the model won't over fit and also will not lead to too much calculate. The number of the neurons of second hidden layer should a little bit larger so that we won't miss the best answer of hidden layer pruning. The hidden layer will use tanh method as activate function and the output layer will use sigmoid as activation function whose meaning is correspond to the represent of the output. The loss function is Binary Cross Entropy which is perform great on dichotomy problem and we will us the SGD to train the model with loss. The learning rate of 0.2 which can let the model keep stable at around 10000 times training and will not over-fitting.

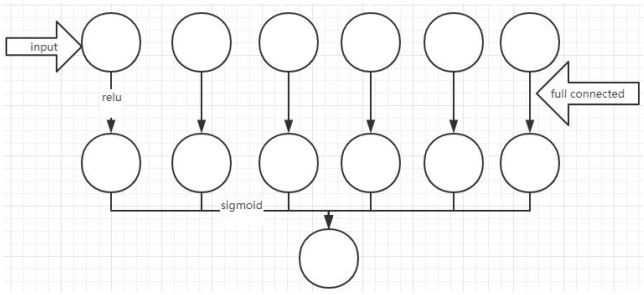

Figure.3 model structure(hidden layers are full connected not shown here)

2.2 Evolutionary Algorithm

Different from traditional gradient descent way to train the model, evolutionary algorithm focus on the gene in the population pool. Whether a gene can be kept for inheritance to generate a new gene generation is decided by the fitness function we set. Also we can control the gene mutation rate and other data during training. Mathematically, the traditional gradient descent way can be seen as a small ball roll down the hillside , evolutionary algorithm is like we scatter many balls and let them start randomly in the hyperspace.The balls can give us a subspace that contains the most suitable balls in the last movement and will improve the probability for you to reach a place with better performance(less loss)[8].

Judging the all parameters as the gene types and our target is to find a combination of parameters to perform on the anger data set as well as possible. The way to arrange the gene is showed in the Table.1

Table.1 Connection between gene and parameters of the model

index	0~35	36~41	42~77	78~83	84~90	91
Meaning	First hidden layer weights	First hidden layer bias	Second hidden layer weights	Second hidden layer bias	Output layer weights	Output layer bias

Initialization

The size of the gene pool is set to 100 and the first generation is generated randomly by the computer. We choose tournament selection as the select strategy and comparing to Stochastic Tournament and roulette wheel selection, tournament selection is more stable and perform better. Then we set the tournament size to be 4 which means that we will choose the gene with highest fitness rate in 4 random selected gene types and add it into next generation.

The way we crossover is blend crossover which means the offspring will has the gene that in the range of its parents. The formula to calculate the offspring is Equation 2. α is always set to 0.5. And the probability of crossover in the experiment is set to 0.5.

$$\text{children} \subset [parent1 - \alpha(parent2 - parent1), parent2 + \alpha(parent2 - parent1)] \tag{2}$$

In the mutation part, the gene will mutate randomly with a probability of parameter 'indpb' which is set to 0.2 here. And the gene(one number of the whole gene type) will change to random number selected from the Gaussian distribution with 'mu'=0 and 'sigma'= 0.5.

Considering the accuracy of the gene type is associated with the the fitness of the gene type to the model. We are using the accuracy as the fitness function of the evolutionary algorithm. Thus the gene that can get high questions accuracy means the corresponding parameters can detect posed anger and genuine anger better and should have a bigger probability to inherit to next generation.

Parameters details

We use the code algorithms.eaSimple() to generate our evolutionary model.The method contains the following parameters:

Population : A list of individuals

Toolbox: A class 'deap.base.Toolbox' that contains the evolution operators

Cxpb = 0.8: The probability of mating two individuals

Mutpb = 0.4: The probability of mutating an individual

Ngen = 500: The number of a generation

2.3 Hidden Layer Compression and Pruning

Implement the model and train them with two different methods which are coming from two totally different inspirations, we are using hidden layer pruning technique in this two different models to not only build two valid models with more concise structures but also trying to analyze the deep connection between gradient descent and evolutionary algorithm like the difference between the same neuron in their hidden layers.

Description

How dose hidden layer compression work? Cutting the neurons in the hidden layer randomly can not improve the performance of the model. As analyzed above,the PCAd1 in anger data set seems much more important than the other component. Reducing the neurons that connect to the PCAd1 will have totally different result with that of reducing the neuron connect to mean component.

The way we reduce the neurons is mainly depend on the activate matrix which is the output matrix of the hidden layer when we feed all training data into the model[2]. Under this circumstance, each column of the matrix is correspond to a neuron. We can then analyze and compare their similarities with each other and the more similar two vectors are, the more two neurons can replace each other.

Finding Neurons

As said before, we are using all input to get the activate matrix of the target hidden layer which is the second hidden layer here and then we are going to calculate the cos similarity by using the method cosine_similarity. Before me do the calculation we should cut the negative part of the matrix and Relu function is just fit this target. Equation 3,4.

$$Firsthidden = \text{Re}lu(input \times firsthidden\_weights + firsthidden\_bias) \tag{3}$$

$$Secondhidden = \text{Re}lu(firsthidden \times senconhidden\_weights + \text{sec}ondhidden\_bias) \tag{4}$$

Then we will analyze the Secondhidden activate matrix. We will test the cos similarity of each columns of the matrix one by one and record the two columns which represent two neurons with highest similarity. The similarity is calculated by equation 5[7].

$$Similarity = \frac{Vector_i \times Vector_j}{|Vector_i| \times |Vector_j|} \tag{5}$$

We will record the index of the vectors with highest cos similarity and reduce one neuron while modify the other.

Reducing Neurons and Modify

The way we reducing neurons also can be called pruning. We can compress the input by reducing the number of the hidden layer. And in order to escape from the situation that the reduced neuron contains important information that we ignore, we have to modify the rest neurons even though we have found a very similar neurons in the same layer. The way to modify the left neuron here is to set the neuron parameters to be the mean of the two similar neurons.(Equation 6,7)

$$Left\_Neuron\_Weights = Mean \cdot (left\_Nruron\_Weights + Reduced\_Neuron\_Weights) \tag{6}$$

$$Left\_Neuron\_Bias = Mean \cdot (left\_Neuron\_Bias + Reduced\_Neuron\_Bias) \tag{7}$$

We will re-train the model firstly and make sure the model is retrain and stable. The we will reduce one neuron of the target hidden layer and train 1000 more times to see the change of the accuracy of the model. We will reduce the neuron one by one to find out the most concise structures of the model.

3 Result and Discussion

3.1 Result based on Gradient Descent

First we are train the model with traditional gradient descent with and without data processing.

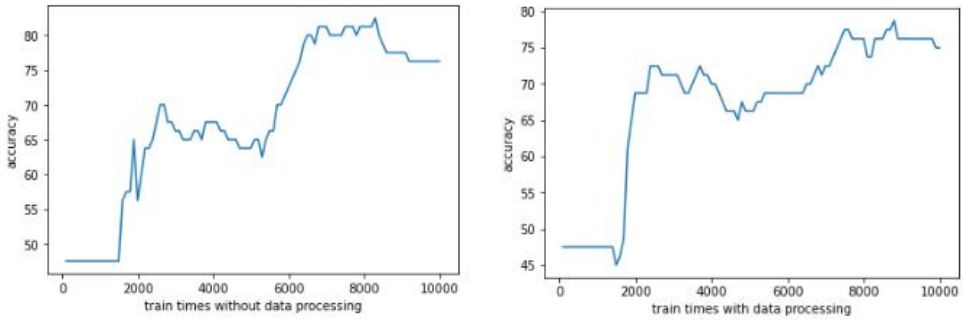

Figure.3 Training with and without processing

The final result is almost the same, both are around 80 percents and the data without processing even shows a better performance than processed data. The reason why this happen might because that the imbalance of the anger data can be eliminate by thousands of training. The speed of the model restrain can partly prove our guess. The processing data make the model restrain faster at around 2500 times training while the other reach 75 percents accuracy after around 6000 times training.The model has spend a lot of time to eliminate the bad influence of the large input data.

As for the activation function of the hidden layer, Tanh and Relu both preformed as below. The highest accuracy and stability of model using Relu is a little bit better so that we choose relu as our activation function.

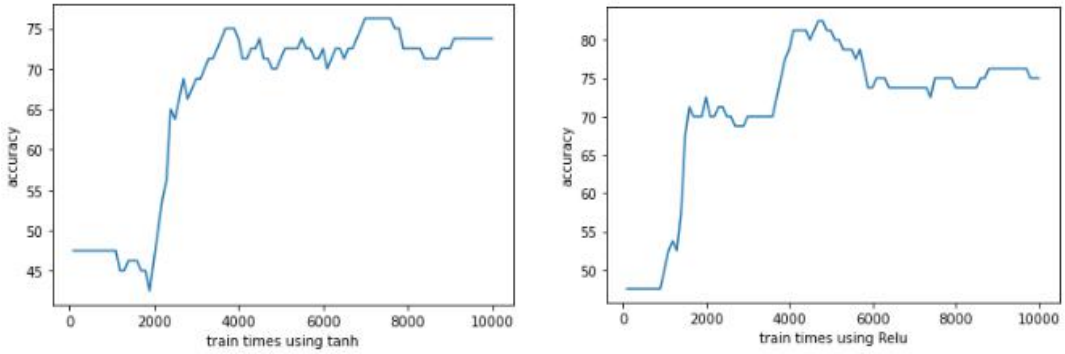

Figure.4 Training with tanh and relu

3.2 Result of Evolutionary Algorithm

The result of the training model with evolutionary algorithm is showed below. Randomly initial the parameters of the model--gene suppose to lead to a 50 percents accuracy while our model start with about 58 percents. This might because of the number of testing data is too small. Then after around 200 generations, our model will reach a high accuracy around 80 percents and will keep stable after that. The best individual which is also our ideal parameters of our model will finally reach about 81percents. The evolution algorithm shows it greatness in speed and stability comparing with the traditional gradient descent training using the same model structures.

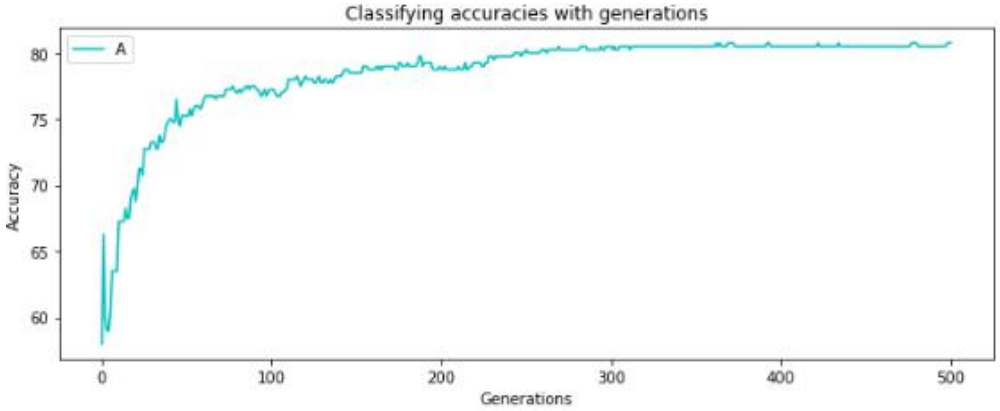

Figure.5 Accuracy of Evolutionary Algorithm

3.3 Progressive Compression/Hidden Layer Pruning

The further test of progressive compression results are showed below. We trained the model 10000 times to make sure the model is almost retrain and have a valid accuracy. Then we reduce the neuron of the hidden layer one by one and retrain 1000 times to let the model get used to the new structure. And from the accuracy changing , we can find that until we reduce 3 neurons the performance of the model is great and pruning even improve the accuracy that the model can detect posed anger and genuine anger. However when we reduce 4 neurons, we performance of the model appear a dramatically dropping. Thus we can draw a conclusion that the most concise model structure based on origin model we design at the start has 3 neurons in the hidden layer and can reach about 77 percents accuracy in detecting posed anger and genuine anger.

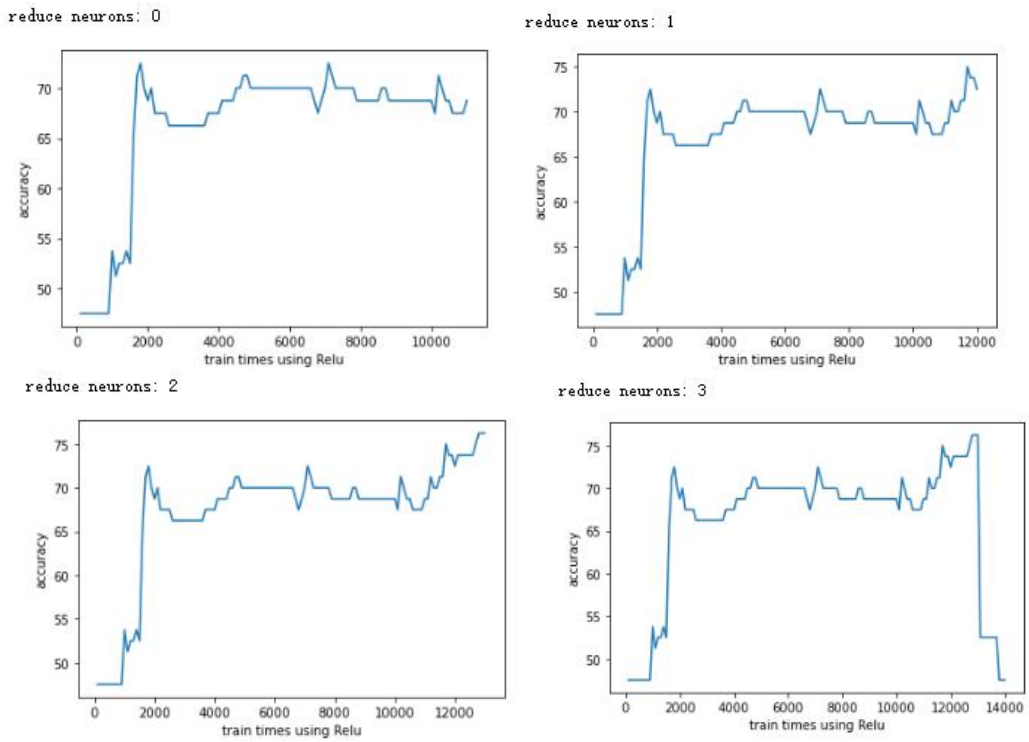

Figure.6 Change of accuracy with hidden layer pruning

3.4 Test the Compression under Evolutionary Algorithm

We try the same compression ratio under the evolutionary algorithm by reducing the same number of neurons of the hidden layer of the model. Comparing with the accuracy, we can see whether the model structure is suitable for the

tasks. And how many neurons should there keep for the model. From the data showed below we can find that the accuracy of the model training with evolutionary algorithm don't change much with the reducing of neurons of the hidden layer. The reason might be that the evolutionary algorithm can restrain with a high speed and the progressive compression can be special that the information of it can not be used in another task(training with evolutionary algorithm).

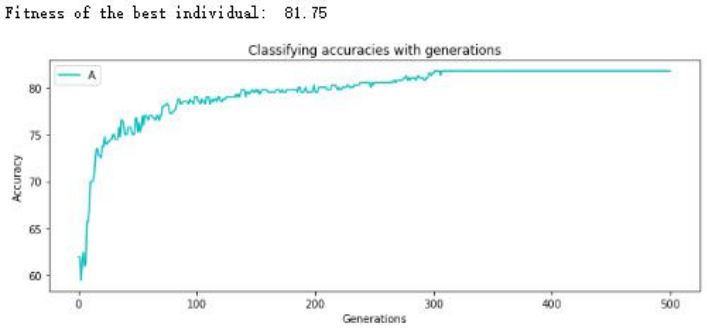

Figure.8 Accuracy of reducing 2 neurons of the hidden layer

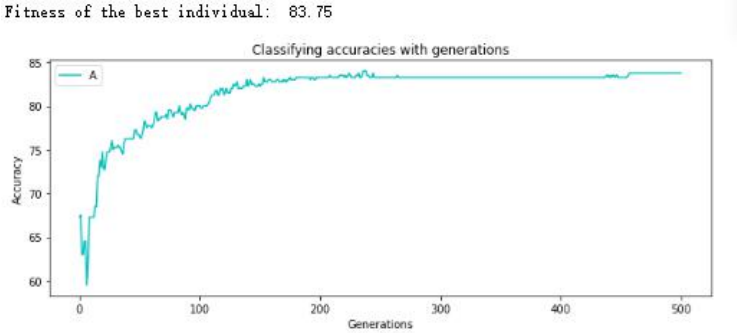

Figure.9 Accuracy of reducing 3 neurons of the hidden layer

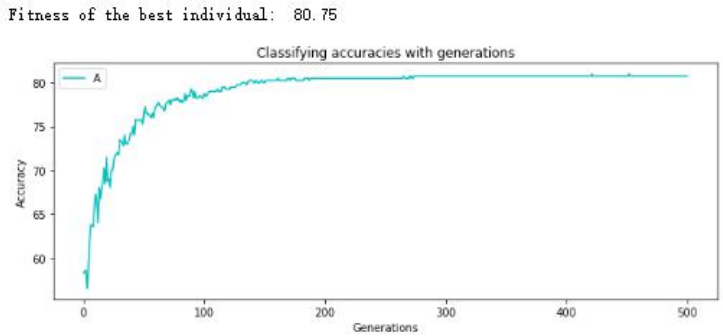

Figure.7 Accuracy of reducing 4 neurons of the hidden layer

3 Conclusion and Discussion

We generate a model that can detect posed anger and genuine anger with an accuracy of over 80%. And also we generate the model with same structure with another way of training. The model using evolutionary algorithm can reach an 80% accuracy too. We then use the progressive compressive and reduce the number of neurons of the hidden layer from 6 to 3 and keep the accuracy of the model to be 80%. However the simplest model we got from progressive compression on gradient descent is not meaningful to the task of using evolutionary algorithm. The model performs well when we reduce the number of neurons of the hidden layer to 2 which is unbearable for gradient descent.

Comparing to the gradient descent training, evolution algorithm performed better in stability and speed. That is because the evolutionary algorithm can be seen that many test have been take place at the same time. Also this algorithm is harder to be trapped in a local minimum. However the highest accuracy are almost the same which means that two models have reach a good result under such a model structure. Comparing to the 99.6% accuracy in [1], 80% accuracy is not good enough. The reason why this happen might be the structure of the model is too simple which may lead to over fitting and the number of data we can use is too small. Whether they are trapped in local minimum is hard to tell.

As for the hidden layer pruning, the result shows that the progressive compression is a good way to reduce neurons and simplify the structure of the model. The result shows a great future of this technique can lead our future research. The simplest model structure we got that performed well can not show its greatness in our experiment. The reason might be that the neurons represent differently in different tasks even they have the same input and output. Maybe analyzing the activation matrix can help us understand the meaning of neuron better and find the deep relationship between different neural network.

There are still some way to improve the model. In [4], Levenberg-Marquardt training function is used in implementation and the adjustable step size might can improve the performance of the model. In [5] , the larger number of neurons of the hidden layer also lead to a better performance.

4 Future Work

We have generate a model based on the anger data set and the performance is not bad. The way of analyzing the data and connect it to the neural network is one of the most important experience we got. Using neural network in detecting other human emotions by using human behaviors data can be a promising work. For example, we can connect speech emotion with whether their facial emotion is genuine[6]. Also we can focus on the pruning skills to find the common feature of hidden layer pruning for different tasks. This may be an important area to optimize the model and improve the performance of neural network when facing too many parameters.

5 Reference

1. Chen, L., Gedeon, T., Hossain, M. Z., & Caldwell, S. (2017, November). Are you really angry?: detecting emotion veracity as a proposed tool for interaction. In Proceedings of the 29th Australian Conference on Computer-Human Interaction (pp. 412-416). ACM.
2. Gedeon, TD, Harris, D, “Network Reduction Techniques,” Proc. Int. Conf. on Neural Networks Methodologies and Applications, AMSE, San Diego, vol. 2, pp. 25-34, 1991.
3. T. D. Gedeon and D. Harris, "Progressive image compression," [Proceedings 1992] IJCNN International Joint Conference on Neural Networks, 1992, pp. 403-407 vol.4, doi: 10.1109/IJCNN.1992.227311.
4. Moré J.J. (1978) The Levenberg-Marquardt algorithm: Implementation and theory. In: Watson G.A. (eds) Numerical Analysis. Lecture Notes in Mathematics, vol 630. Springer, Berlin, Heidelberg. https://doi.org/10.1007/BFb0067700
5. Md Zakir Hossain and Tom Gedeon. 2017. Classifying Posed and Real Smile from Observers’ Peripheral Physiology. 11th International Conference Pervasive Computing Technologies for Healthcare (PervasiveHealth '17), EAI Conference Series,Barcelona, Spain, 1-4. Retrieved from http://bit.ly/2gkllCZ
6. Han, Kun / Yu, Dong / Tashev, Ivan (2014): "Speech emotion recognition using deep neural network and extreme learning machine", n INTERSPEECH-2014, 223-227.
7. Gedeon, T.: Indicators of hidden neuron functionality: the weight matrix versus neuron behaviour. In: Proceedings 1995 Second New Zealand International Two-Stream Conference on Artifificial Neural Networks and Expert Systems, pp. 26–29. IEEE (1995)
8. Korolev, L.N. Evolutionary computations and neuronet and genetic algorithms — formal statements. J Math Sci 168, 80–88 (2010). https://doi-org.virtual.anu.edu.au/10.1007/s10958-010-9976-z