Training a Neuron Network with Limited Experiment Data and Hidden Neurons

Ya Ting Wang
Australian National University
Canberra, Australia
Yating.Wang@anu.edu.au

***Abstract*—Pupil diameter can reflect human response to observed environment [1]. We can identify people's response by analyzing the change of pupil diameter. In this report, we aim to use a fully connected multilayer perceptron (MLP) with limited experiment data to identify whether people watched a fake smile video or a real smile video. The experiment data is recorded from 10 Asian observers (6 males and 4 females), while watching 9 real smile stimuli and 10 fake smile stimuli [2]. We use a generative adversarial network (GAN) to help augment the training data. Furthermore, in order to reduce the training time of the MLP, we prune some hidden neurons based on the cosine similarity. The results firstly indicate that the increasing training data helps reduce the loss eventually. The results also indicate that repeated pruning will gradually decrease the performance of the neuron network.**

Keywords—multilayer perceptron, MLP, resample, pupil diameters, generative adversarial network, GAN, classification problem, distinctiveness measure, pruning, pupil diameter, limited experiment data

I. Introduction

Dataset augmentation is a popular technique applied in visual recognition tasks. People have successfully utilized related techniques to construct competition winning classifiers these years. Using data augment techniques to increase the training examples, which improves the performance in imbalanced class problems [3]. Although dataset augmentation is not easily used in all domains, some other researchers have successfully applied data augmentation to singing voice detection. By adding Gaussian noise to the input and some operation on the audio signal, they can improve model performance [4].

Similarity measure is another popular technique to distinguish the redundant or less important hidden neurons. With this technique, we can prune the hidden neurons with a more precision way, which will eventually help save computational space and resources.

In this report, we firstly use the data augment technique to increase the training examples. Due to the limited experiment dataset collected by from only 20 samples of pupil size records, we construct a simple generative adversarial network to generate numeric dataset with similar attributes to the real data. And then we use a fully connect MLP to identify what type of video the participant has watched. During the training process, by using the distinctiveness angular measure, we gradually reduce the redundant or less important hidden neurons to investigate the relationship between the size of hidden layer and neuron network performance.

II. Methods and Data

A. Datasets

The pupil dataset was used for the experiments. There are two classes, the real smile and the fake smile. This dataset consists of 12 samples. Each sample has two subsamples, labelled with 'real' or 'fake'. Each subsample contains 540 pupil sizes. These pupil data of each subsample are collected in time 60s. The training stage uses the first 5 samples and the validation stage uses the second 5 samples. The testing stage uses the last two samples.

B. Resampling

Data resampling provides a collection of techniques. These techniques can be applied to balance or better balance the class classification. However, in this report, we use resampling techniques to split the training set into small samples with the equal size.

The purpose of resampling is to increase the number of training samples. The provided dataset has a balanced class distribution. However, it only contains 12 samples. Before resampling, each sample has 541 pupil diameters. If we didn't reduce the size of each sample, the size of neurons of input layer would be very large. The lager number of neurons of the input layer will lead to a large number of weights, which will lead to overfitting. Furthermore, it is also hard to set batch size at the training stage since we only use 10 samples during the training stage. Therefore, it could be helpful if we resampled the dataset. Fig 1 shows the process of resampling. We take one data from every 10 data to construct a sample. Each new sample has 10 pupil diameters. Since we didn't change the class label of the new samples, the new samples have a balanced class distribution as well. After resampling, the training samples increase from 10 to 540 respectively.

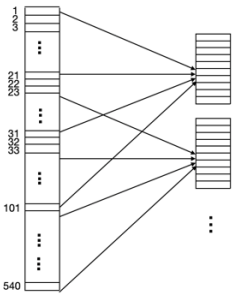

Fig.1. Resampling

C. Dataset augmentation

We use a simple generative adversarial network (GAN) to augment the training samples (see Fig. 2). GAN generates new samples based on the dataset after resampling. The samples generated by GAN mimic the pupil diameters recorded in a

fixed time. These new samples look like $[[x_1, x_2, \ldots, x_{10}], [y_1, y_2, \ldots, y_{10}], \ldots]$. We use dataset labelled 'real' to generate 'real' samples. We also use dataset labelled 'fake' to generate 'fake' samples. We export generated 'real' samples and 'fake' samples to data-real.xlsx and data-fake.xlsx respectively. The related source code can be found in ProduceFakeData-GAN.py.

Because of the complexity of the generative adversarial network, we didn't use some validation techniques to find the optimal hyperparameters. We directly use the hyperparameters according to the thumb of rules from a researcher [5].

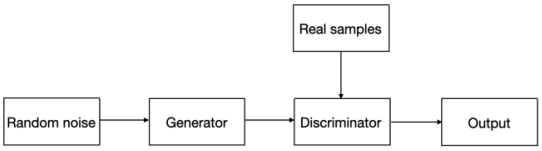

Fig.2. Generative adversarial network architecture

- *Step-1 generating random noise.*

  Generate some random noise as input for the generator network.

- *Step-2 constructing a generator neuron network.*

  Fig 3 shows the architecture of the Generator neuron network. Generator neuron network accepts the random noise as input to generator fake data.

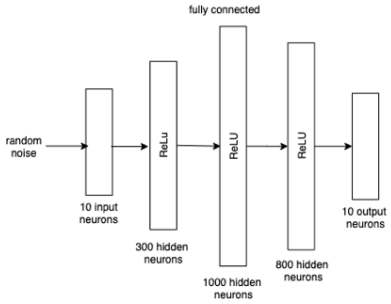

Fig.3. Generator neuron network architecture

- *Step-3 constructing a discriminator neuron network.*

  Fig 4 shows the architecture of the Discriminator neuron network. We use binary cross-entropy as its loss function since Discriminator conducts a binary classification task. Stochastic gradient descent is used as its optimizer. Discriminator neuron network firstly accepts the real samples and then it accepts the data generated from the generator neuron network. Discriminator will do the back propagation for all the input data.

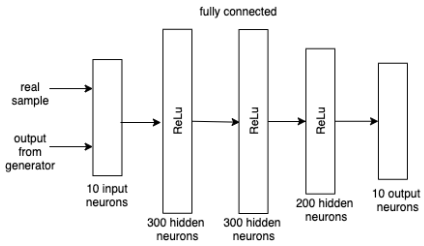

Fig.4. Discriminator neuron network architecture

- *Step-4 exporting the generated data to files.*

  The generated data is exported to data-fake.xlsx and data-real.xlsx respectively.

- *Step-5 removing the outlier data from the generated data.*

  With the process of training, the quality of data generated by GAN will be gradually improved. We removed around 300 samples generated at the early stage.

- *Step-6 comparing the generated data with the real samples.*

  Fig 5 shows the 'real' labelled samples generated by GAN (left) and the original samples (right) with the same label. The range of pupil diameters generated by GAN is from 0 ~ 0.5. Compared with the original samples, the pupil diameters from 0.5~1.0 fail to be generated by GAN.

  Fig 6 shows the 'fake' labelled samples generated by GAN (left) and the original samples (right) with the same label. The range of pupil diameters generated by GAN is from 0 ~ 0.8. Compared with the original samples, the pupil diameters from 0.2~1.0 fail to be generated by GAN.

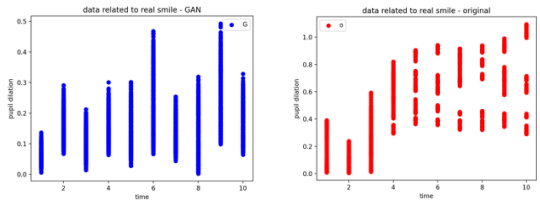

Fig.5. 'real' samples from GAN and original samples

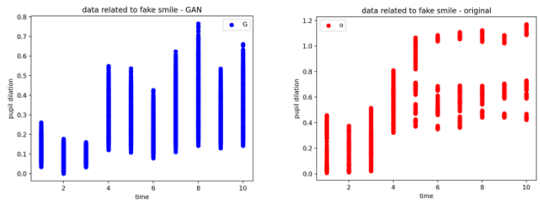

Fig.6. 'fake' samples from GAN and original samples

From the above diagrams, we think that the generated data is similar to the original samples. Therefore, we use these generated data to train the model.

D. Multilayer perceptron classification

We designed a fully connected multilayer perceptron classification with two hidden layers. Since it is a classification task, we use sigmoid as the activation function. In addition, we use Adam as the optimizer since Adam performs the best on average. We choose cross entropy as the loss function. The output is 0 or 1 (see Fig. 7). "0" represents that the participant watched a fake smile video. "1" represents that the participant watched a real smile video, as in:

$$y(x) = Sigmoid(f(f(x, w_1), w_2), w_3) \quad (1)$$

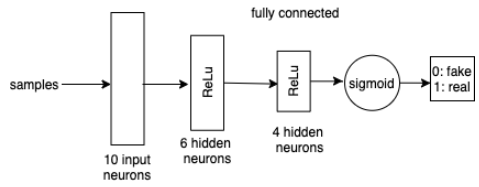

Fig.7. Multilayer perceptron classification architecture

E. Validation

Since the amount of original data is small, we use the data generated by GAN to augment the validation set. Instead of splitting the validation set into several subset, we directly use the validation set to find out the optimal hyperparameters. There are two models to be validated. The first model has 4 layers and the second model has 3 layers. Table 1 shows that these two models have the same accuracy and performance. Therefore, we choose the second model since the second model has less layers and neurons, which means it saves the computation resources. Since the number of the second model is very few, we think it is good enough to use these numbers as the hyperparameters.

TABLE I. SPCIFICATION OF TWO NEURON NETWORK

	Number of neurons	***accuracy***
1^{st} layer	50	97%
2^{st} layer	30	
3^{st} layer	20	
4^{st} layer	2	

	Number of neurons	***accuracy***
1^{st} layer	6	97%
2^{st} layer	4	
3^{st} layer	2	

F. Repeated pruning

According to the technique paper [5], by measuring the distinctiveness of hidden neurons, we can firstly compute the cosine similarity between two neurons and then we remove the neurons if the angle between two vectors is too small. According to the following equation, the more the degree of two neurons are similar, the closer the value of $\Phi(v_A, v_B)$ is to 1.

$$\Phi(v_A, v_B) = \cos^{-1} \frac{v_A * v_B}{\|v_A\| * \|v_B\|} \qquad (2)$$

The removing will not dramatically affect the result of a neuron network. In order to verify the relationship between cosine similarity of neurons and accuracy, we recorded the minimum cosine similarity when removing a neuron from the first hidden layer (see Table. 2).

TABLE II. NEURON SIMILARITY OF THE FIRST HIDDEN LAYER

similarity	***n2***	***n3***	***n4***	***n5***	***n6***
n1	0.29	0.26	0.07	-0.3	-0.5
n2	-	0.04	-0.08	-0.29	-0.19
n3	-	-	0.23	0.20	-0.04
n4	-	-	-	0.13	-0.02
n5	-	-	-	-	0.9

The author also uses neuron network which is repeated pruned to analysis the relationship between the number of hidden neurons and the neurons behaviors. Following this experience, we use the similar approach to verify the relationship between the number of neurons and neuron network performance. We repeated prune the units of the firstly hidden layer. After removing a neuron, the network will be retrained for 2 epochs. The performance is getting down after repeated pruning.

III. RESULTS AND DISCUSSION

A. Augment training data imporve the performance of the neuron network

We use generated data to augment the training samples. Table 3 shows the loss and accuracy before and after using data augmentation. We can find that the accuracy at the testing stage dramatically increases after using data augmentation.

TABLE III. COMPARE LOSS AND ACCURACY BEFORE AND AFTER DATA AUGMENTATION

	Before augmentation	***After augmentation***
Training loss	*0.6%*	*0.3%*
Testing accuracy	*50%*	*97%*

Due to the limited knowledge and experience, we failed to figure out why there are some unexpected loss change during the training stage (see Fig. 8). The following diagram shows that there are some outliers beyond the main trend. We guess it may be caused by the data generated by the GAN. As described in Fig.5 and Fig.6, although the trends of the data generated by GAN and the original sample are slightly similar, there still exists some differences between these two types of data.

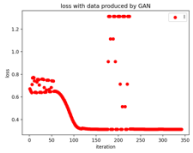

Fig.8. Loss change during the training stage

B. Repeated pruning gradually decrease the the performance of the neuron network.

We firstly compute the cosine similarity between neurons on the first hidden layer. And then we gradually prune the neuron which is similar to other neurons.

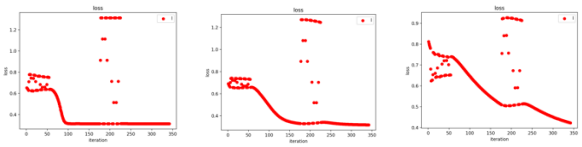

Fig.9. Repeated pruning

The above diagrams show the loss getting to decrease in different situation. The x axis represents the number of epochs. The y axis represents the number of loss.

The first diagram shows the loss getting to decrease when we didn't prune any neuron of the first hidden layer. The loss quickly decreased at around second interval.

The second diagram shows the loss getting to decrease when we prune the last neuron of the first hidden layer. The reason we prune this neuron is that the cosine similarity between neuron 5 and neuron 6 is 0.9 (see Table. 1), which means these two neurons are cosine similar. Compared with the first diagram, it is obvious that the loss getting to decrees is slower than the first one. The loss costs around double epochs to get to the lowest loss.

The third diagram shows the loss getting to decrease when we continued to prune the fourth neuron of the first hidden layer. We choose to prune this neuron is that the cosine similarity between fourth and third neuron is 0.23, which is almost the highest one among other values (see Table. 1). Compared with the first diagram, the loss getting to decrease is much slower than the first one. It costs around triple epochs to get to the lowest loss. Besides, the loss at the early epoch is almost double compared with the first diagram.

In conclusion, repeated pruning decreases the performance of the neuron network.

CONCLUSION AND FURTHER WORK

In our work, we explore a way to augment the training samples. We firstly split the original dataset into subsets. And then we use these subsets to generate more training samples by using a simple generative adversarial network. At the next stage, we construct a fully connected multilayer perceptron classification. After validation, we choose an optimal model. And then we compute the cosine similarity between neurons of the second hidden layer. By pruning three neurons step by step, we reach a conclusion that repeated pruning decreases the performance of the neuron network.

Although we get a high accuracy at the testing stage, there are some questions still needed to be answered. Why there are some unexpected outliers during the process of loss decreasing. Is there any way to improve the quality of data generated by GAN? How can we measure the new sample's quality of after resampling?

Out further work will focus on two points. The first one is to improve the performance of GAN. The second one is to find a more mature way to resample the original data.

REFERENCES

[1] Hossain, Md & Gedeon, Tom & Sankaranarayana, R & Apthorp, Deborah & Dawel, Amy. (2016). *Pupillary Responses of Asian Observers in Discriminating Real from Fake Smiles: a Preliminary Study*.

[2] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. 2002. *SMOTE: synthetic minority over-sampling technique*. J. Artif. Int. Res. 16, 1 (January 2002), 321–357.

[3] Schlüter, Jan & Grill, Thomas. (2015). *Exploring Data Augmentation for Improved Singing Voice Detection with Neural Networks*.

[4] Tanay A. (2019). Train your first GAN model from scratch using PyTorch. Retrieved from https://blog.usejournal.com/train-your-first-gan-model-from-scratch-using-pytorch-9b72987fd2c0

[5] T. D. Gedeon, "Indicators of hidden neuron functionality: the weight matrix versus neuron behaviour," *Proceedings 1995 Second New Zealand International Two-Stream Conference on Artificial Neural Networks and Expert Systems*, Dunedin, New Zealand. (1995). pp. 26-29, doi: 10.1109/ANNES.1995.499431.