Application of Bimodal Distribution Removal and Recurrent Neural Network (RNN) to Neural Network for Human Eye Gaze Pattern Recognition

Lei Wang

Research School of Computer Science

The Australian National University, Acton, Canberra, ACT0200, Australia

U5947698@anu.edu.au

Abstract

Human eye gaze patterns for observing different objects vary a lot. With the differences of the patterns, the logical process of humans make can be identified. Therefore, investigating the relationship between human eye gaze patterns is one of the popular topics in machine learning. Compared to existing method which potential data imbalance in the dataset. Therefore, Bimodal Distribution Removal (BDR) is introduced to a two-layer simple linear neural network. The BDR can remove outliers in the training data. Therefore, it is widely used in machine learning. However, it is turned out to have negative impact on the investigation of human eye gaze patterns, which also indicates that there is no imbalance within the given datasets. Therefore, a new method is developed by implementing Recurrent Neural Network(RNN) approach. Since RNN can reveal the internal relationship between previous output and current input by inputting the previous output to the current input, it can mimic the internal relationship between human logic processes at each time step. The RNN approach turns out to be helpful for improving the performance. However, due to the nature of RNN, the speed of training the model is much slower than simple 2-layer neural network.

1. Introduction

Neural network applications cover a lot of areas in our daily life. One of the most popular areas is investigating human visual behaviours. In the investigation of relationship between human visual behaviours logical processes, it was found by previous research that human eyes and their cognitive processes are highly correlated to each other. Therefore, this finding enabled us to build a model based the relationship between human eyes patterns and their cognitive processes, which was completed by Dingyun Z, B. Sumudu U. Mendis, Tom Gedeon, Akshay Asthana, and Roland Goecke[1]. According to the work by Dingyun Z, B. Sumudu U. Mendis, Tom Gedeon, Akshay Asthana, and Roland Goecke[1], they successfully established a model by correlating human eye gaze pattern with human cognitive processes. They introduced the hybrid fuzzy approach: hierarchical fuzzy signature construction with Levenberg-Marquardt learning of the generalized Weighted Relevance Aggregation Operator (WRAO) for modeling recognition of human eye gaze patterns between face scanning and text scanning.

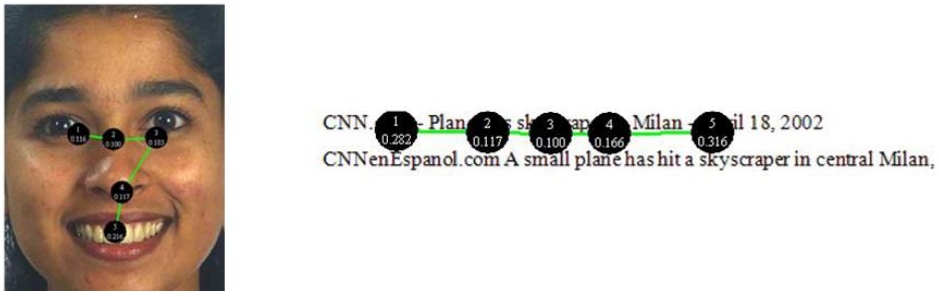

Figure 1.Five Fixations for Data Collection of Eye Gaze Patterns (Source:[1])

The way that they collected is as follows: by researching, it was found that people concentrate their gaze fixations onto the interesting or informative regions. Therefore, by measuring the distances

between 2 points on the pre-set 5 points on human face, as figure 1 shows below, the model can be constructed. Based on the data collected, a new model is built by introducing RNN to a simple two-layer neural network. After my previous experiments of introducing Bimodal Distribution Removal (BDR) to the model, it was found that there was no improvement of the training and testing accuracy. Also, the error distributions of the models with and without BDR were very similar. Therefore, it can be concluded that the data set is balanced, for which the BDR part is not included in the latest model anymore.

2. Method

In this section, architectures of two-layer simple linear networks (with and without Bimodal Distribution Removal) and RNN networks (one-layer and two-layer RNN) are explained. Compared to the simple linear neural networks, RNN is used and BDR is eliminated from the model. To investigate how effective RNN is for improving the accuracy and how much RNN is affecting the training time.

2.1. Construction of the Two-layer Neural Network without and with BDR

The neural network model consists of the following parts: input neurons, hidden neurons and output neurons. The numbers of input and output neurons are set to be the size of training input, which are 4 and 2 respectively based on the data set; the number of hidden neurons is set to be 6, given that there are 4 input neurons and 2 output neurons; learning rate is set to be 0.01 by past experience; the number of epoch is set to be 400 to collect loss and accuracies every 50 epochs.

In terms of the activation function, sigmoid is selected to avoid below up the activation. While it is turned out that in this neural network, sigmoid and ReLu functions have similar performance.

The Bimodal Distribution Removal method is used to remove potential bias in the data set, which therefore improves the accuracy of the training. The purpose of implementing the BDR method is to remove the potential biasing data points in the training set, such that the neural network can provide a better performance for investigating human eye gaze patterns. The potential bias can exist in some scenarios. For example, when the participants of the experiment unintentionally performed an eye gaze pattern as ‘recognized’ but they actually do not recognize anything for the first several times of data collections. They perform like that simply because they are nervous about the experiment at the beginning. However, these biased data may lead to the neural network incorrectly trained.

However, it is crucial to point out that if the original training set does not have any biased data, the BDR method may not be effective in improving the accuracy of the neural network, which proves to be true after comparing the results to the same neural network without BDR.

According to [2], the implementation of the BDR method is conducted as follows:

1. Starting the training if the variance of errors is in the range of [0.01, 0.1]
2. Calculating the errors by: error= Ypredict – Ytarget
3. Calculating the variance of the errors δ ts
4. Indexing the data with its error > δts to form a subset
5. Calculating the average error of δts as δss and standard deviation σss.
6. Removing the terms from the training set with error $\geq \delta ss + \alpha\sigma ss$, α is set to be 0.5.
7. Repeating steps 2-6 until the variance of errors ≤ 0.01, stop training.

2.2 Construction of the One-layer RNN

The neural network model for one-lay RNN consists of the following parts: input neurons, hidden neurons, number of layers and output neurons. The numbers of input and output neurons are set to be the size of training input, which are 4 and 2 respectively based on the data set; the number of hidden neurons is set to be 5, given that there are 4 input neurons and 2 output neurons; learning

rate is set to be 0.05 by past experience; the number of layers is 1 because it is one-layer RNN; the number of epoch is set to be 400, making it dividable by 50 to collect accuracy every 50 epochs.

RNN is used to output of the model. The output is calculated by the built-in function of Pytorch: nn. RNN. The logic is simple. However, the design of tensor sizes is challenging because the built-in Pytorch function RNN requires a 3-Dimentional input and it outputs 3-D data as well. Therefore, some reshaping operations are required.

Variable of Tensors	Initial Training Input	Initial Training Target	Input of RNN	Output of RNN	Final Output of the Neural Network
Shape	[1, length of the training set, 4]	[length of the training set]	[1, 1, 5]	[1, length of the training set, 5]	[length of the training set, 5]

Table 1. Shapes of Tensor Variables

As table 1 shows, several reshaping operations are completed. Given that the data set has 4 features, 1 target class, and only 169 samples, the input of RNN is set to be [1, 1 (length of tensor), 5 (number of hidden neurons)]. After passing the input to RNN function, the output is a tensor of [1, length of the training set, 5 (number of hidden neurons). Since the loss function (cross entropy) and ReLu function require 2-dimention inputs, the shape of their input tensors should be reshaped into 2-D, which is [length of the training set, 5]. The batch size for one-layer RNN and two-layer RNN is 10

In terms of the RNN algorithm, the RNN algorithm is used to deal with problems that require ‘remembering sequences’, which can be helpful for our model because there may be some correlation between logic processes of human at each time step. The application of RNN to the model is as follows:
1. The built-in RNN function takes the input tensor of training features.
2. By using the previous output as the current input, calculate the final state.
3. The final state is generated as the output. Then it will be compared to the actual training target, the errors will be returned for weights to update.

2.3. Optimizer and Loss Function

Based on previous experience, for loss function, cross entropy is selected because it usually performs better than MSE for classification problems. In the familiarity recognition data set, the output is either 1 or 0, which is a typical classification problem. Therefore, it is suitable to set the loss function as cross entropy.

For optimizer, similarly as the choice of loss function, SGD and Adam perform similarly. Therefore, there is no huge difference for choosing either SGD or Adam particularly and Adam is finally chosen for RNN models.

2.4. Training the model

	Training data set	Testing data set	Validation data set
Number of data	169	153	80

Table 2. Number of Data in Each Data Set

The number of epochs is set to be 500, with X as the training input tensor and Y as the training target tensor. By performing forward pass, we pass the predicted Y values consisting 5 columns to the training module. The predicted Y values are converted to 1-column data set for comparison by using torch max function and torch.nn.functional softmax function. Then, the output tensor is generated. The training loss and accuracy are collected every 50 epochs.

After training, validation is used to validate the trained neural network. Due to limited number of training data set and testing data set, which is 169 for the training set and 153 for the testing data set, it is not enough to perform multiple k-fold cross validation. Therefore, the validation data set is constructed by splitting half of the training data set, which is 80.

Also, to avoid internal dependency of data in the training data, the validation data set is constructed as follows:

1. Shuffling the indices of training data set by using random.shuffle function. Therefore, the original ordered indices of training data set [0, 1, 2, 3……,167, 168] is transformed to a randomized index sequence of list, **valid_index = [37,54, 6, 20,….,12,80,19]**.
2. The validation data set is obtained by indexing the first 80 data of the training data set using the randomized index sequence, which is ***training_data_set[valid_index[0:80]]***

By doing the randomization, the validation data set is obtained randomly. Also, the possible internal data dependency in the original training data set is eliminated. Therefore, despite lack of data, we can treat the validation data set as a 'new' data set, which meets the requirement of validation.

In terms of the specific validation standard, training accuracy and loss are compared to the training. A valid neural network is validated as follows:

1. The training accuracies of the training and validation should not have a huge gap.
2. The training losses of the training and validation should be close to each other.

Additionally, a confusion matrix for training is also displayed to check details of the training. Training process is the same for all models.

2.5. Testing

Testing is completed by inputting the testing data set to the trained neural network. Then, the predicted values of the neural network are compared to the testing targets to check whether the predicted outputs match the testing targets. The testing accuracy is calculated by summing up the total number of correct predictions, then then sum is divided by the total number of testing data points. Testing process is the same for all models.

2. 6. Two-Layer RNN Construction

To further improve the RNN model, another hidden layer is added to the model to investigate whether implementing another layer can improve the model or not and how another hidden layer will influence the results.

Compared to the constructed 1-layer RNN, the new 2-layer RNN is constructed by adding a hidden layer after the model implements RNN algorithm. By doing this, more training can be done and therefore some improvements of accuracy and losses can be achieved. Training, testing, optimizer and loss function

3. Results and Discussion

The results of the neural networks with one-layer RNN and two-layer RNN will be compared and discussed in this section. Also, the two-layer neural network without RNN will be compared to the RNN models to investigate the advantages and drawbacks of RNN.

3.1. Training Time for 3 Models

Since the two-layer neural networks with and without BDR have very close training time, only training time of two-layer network without BDR is recorded.

As table 3 shows, the training time of the 3 models are significantly different. The trainings of the models are completed on a laptop without GPU. The two-layer network without RNN is trained significantly faster than other 2 RNN networks. This is because the two-layer network in assignment 1 was constructed with one hidden layer and one output layer, without any complex

calculations required. In the hidden layer, only an activation is completed, and the hidden output is passed to output layer directly. The output layer is using the built-in torch.nn.linear function. Therefore, the calculation is not complex.

Model	Two-layer network without RNN	One-layer RNN	Two-layer RNN
Training Time of 500 epochs (seconds)	0.69	23.24	43.41

Table 3. Training Time of 3 Models

However, for RNN neural networks, more calculations are required. The algorithm considers all the inputs at the same time and continuously inputs the previous output to the current input. These iterations are therefore time consuming, making the 2 RNN neural networks trained slower than simple linear neural network.

It can also be observed that by adding 1 hidden layer to the one-layer RNN, the training time of the two-layer RNN is almost as much as 1 time of the one-layer RNN. Therefore, to pursue faster model, the less layer that model has, the faster it is trained.

3.2. Loss

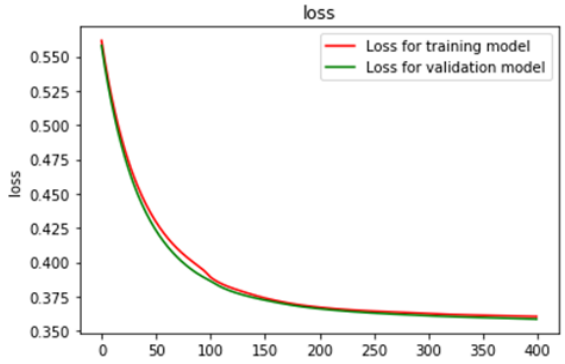

Figure 2. Loss for Two-Layer Simple Linear Model without BDR

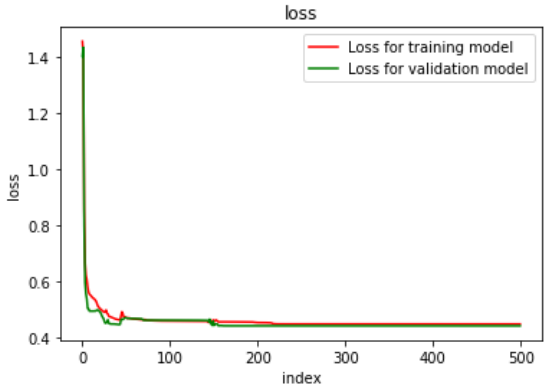

Figure 3. One-Layer RNN Loss

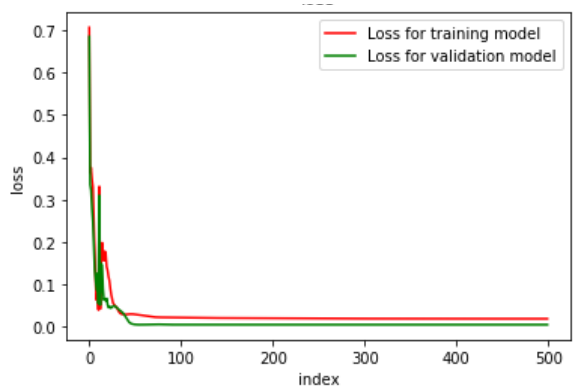

Figure 4. Two-Layer RNN Loss

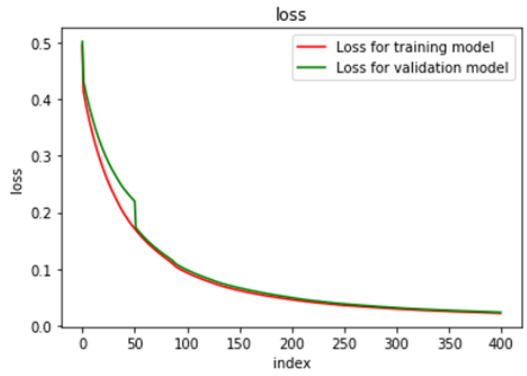

Figure 5. Loss of Two-lay Simple Linear Neural Network with BDR

From figure 2, it can be observed that the loss for both training and validation models are very close. Also, the loss is converged to about 0.36. Figure 5 shows that the loss patterns for training and validation are similar: the losses converge at the similar index; the converged losses are very

close. Therefore, the neural network is valid. Compared to figure 2, the losses of training and validation with the BDR method are significantly smaller than those without the BDR method, indicating the BDR method can significantly reduce the losses for the neural network of human eye gaze pattern investigation. The reason can be that the BDR method eliminates the outliers, therefore, during the training process, less data is lost after the elimination.

From figure 3, it can be observed that the loss of one-layer RNN for both training and validation models and the loss of it is slightly higher than the two-layer simple linear model and it is higher than the two-layer RNN (shown in figure 4). The reasons can be as follows: The one-layer RNN model is overfitted. Even if one layer is added, the loss of one-layer RNN is still significantly higher than two-layer RNN model, but the training and testing accuracies of both models are close. After printing the sizes of the tensors, it is found that the dimension of two-layer RNN input is greater than one-layer RNN. This optimizes the problem of overfitting.

By comparing the losses of the two-layer RNN and the model with BDR in assignment 1 (shown in figure 5), it can be observed that the loss of the two-layer RNN model is smaller. Therefore, a proper RNN can reduce the loss of the model.

3.3. Training, Testing and Validation Accuracies.

It was found out that two-layer simple linear neural network with and without BDR have training, testing and validation accuracies less than 90%, indicating some improvements can be achieved. As figure 6 and figure 7 show, final testing accuracies for both models in assignment 1 is about 84%.

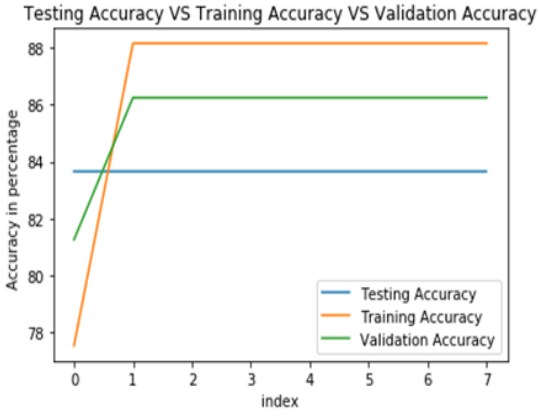

Figure 6. Simple Linear Network without BDR

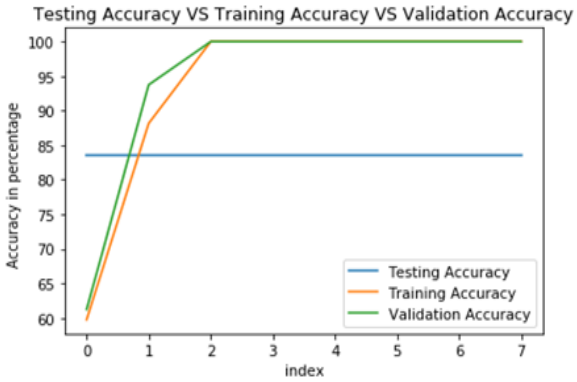

Figure 7. Simple Linear Network with BDR

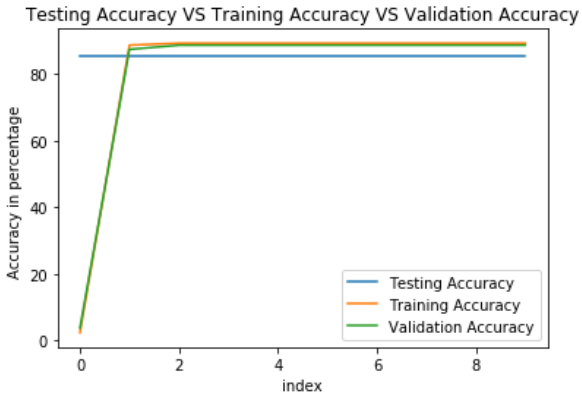

Figure 8. One-Layer RNN Accuracies

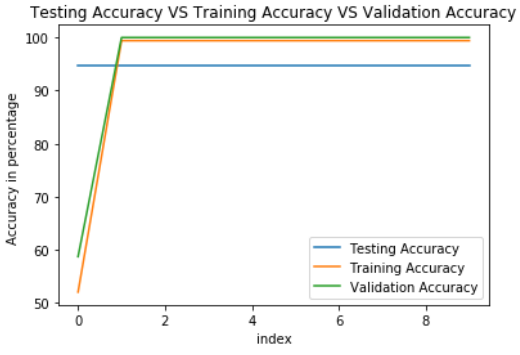

Figure 9. Two-Layer RNN Accuracies

In terms of the testing, training and validation accuracy of the simple linear neural network with and without the BDR method, figure 6 and figure 7 show that the training and validation accuracy are

about 12% higher than the neural network without the BDR. However, the testing accuracy remains unimproved. Therefore, by comparing the accuracies of the neural network with the BDR and without the BDR, a conclusion can be made that the BDR method improves the training accuracy, but it does not improve the testing accuracy. The reasons can be as follows:

1. For the familiarity recognition data set, the data set demonstrates a very weak bimodal distribution. Therefore, removing the bimodal-distributed terms does not improve the neural network very much.

2. The testing data set may be bimodal-distributed, therefore, even if the neural network has removed bimodal distribution from itself, the testing data can still bias the testing results.

3. The training data set does not behave a bimodal distribution. Therefore, removing the 'outliers' will only improve the training accuracy, but testing accuracy remain unchanged.

Models implementing RNN improve to around 86% for one-layer RNN and 95% for two-layer RNN as figure 8 and figure 9 show respectively. Compared to the models not using RNN, the accuracies of RNN models improve by 10%. Therefore, RNN is helpful for improving accuracies. This can be explained as follows: RNN linked previous output and current input, which therefore considers potential internal relationship between datasets. Also, RNN stores all the information of the training data throughout the training process, according to Introduction to Recurrent Neural Network[3], it is able to calculate the weights more precisely and update them more properly. After calculation including this internal relationship as well as updating the weights, the accuracy is improved. Therefore, RNN is helpful for improving accuracy of the model.

3.4 Error Distribution of Two-layer Simple Linear Neural Networks

The main purpose of implementing BDR is to remove potential outliers in the given datasets. Therefore, error distribution of the models with and without BDR should be analysed.

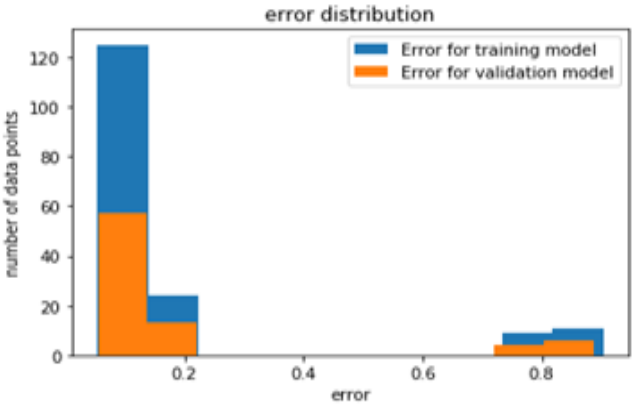

Figure 10. Error Distribution of Training and Validation without BDR

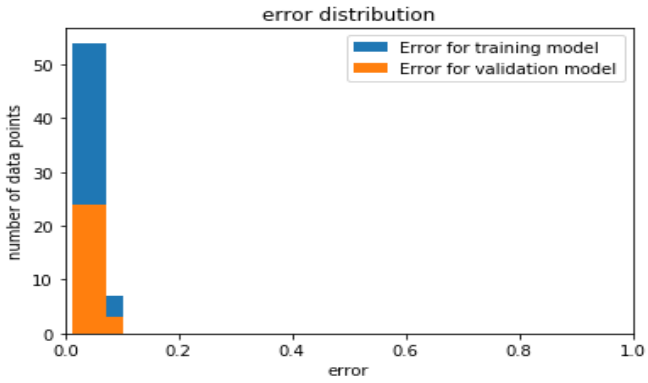

Figure 11. Error Distribution for the Neural Network with the BDR

As figure 10 shows, the error distribution of training and validation perform similar pattern, where the errors of the both are mainly centred at about 0.1 with minority of errors at about 0.8. Also, the numbers of the errors at each value have similar portion of the total errors correspondingly. By comparing the training and validation, it can be observed that the number of validation errors centred at 0.1 is about half of that of the training and this is also true at 0.8, which matches the fact that the number of validation data set is half of the training. Therefore, by observing the error distribution, the validation passes the training. The neural network is valid. In addition, there are a number of errors at 0.8, which indicating that 0.8 is the second 'peak' of the bimodal distribution. Therefore, it is possibly helpful for introducing the BDR method to improve the network.

From figure 11, it can be observed that error distribution patterns for training and validation perform similarly, which are both centred at about 0.02 (to better visualize the errors, the widths of

the bars in figure 5 are expanded, the actual values are about 0.02). Therefore, the neural network is valid. Moreover, there is currently no error at about 0.8 compared to figure 2, indicating that the BDR method successfully removes the outliers from the training data set.

However, even if BDR removes the outliers in the dataset, it still cannot improve the accuracy. Therefore, BDR is not helpful for improving the model in this project.

3.5 Comparing to the Technique Paper

According to the technique papaer[1], the training accuracy is 80.23% and the testing accuracy is 80%. Therefore, both simple linear network and RNN can improve the performance compared to the model built in the original technique paper.

4. Conclusion and Future Work

In conclusion, the BDR method can improve the training performance of the neural network by 20% of accuracy. However, it cannot influence the testing accuracy of the neural network. Therefore, the BDR method is not suitable for the improvement of the neural network, as it does not improve the predictive performance; compared to two-layer simple linear networks with and without BDR, RNN can improve the accuracy of the neural network by 10% and the loss of the model can be reduced if a proper RNN is constructed. However, training of RNN takes significantly longer time than simple linear neural networks. Since the size of provided data set is small, which has only 169 samples, RNN can still be used. If datasets are very big, it is a difficult task to train the model. RNN requires significantly more on hardware than simple linear networks. In summary, RNN can be a choice for investigating eye gaze pattern because it is more accurate, if hardware is good enough and there is enough time.
For future improvements, genetic algorithms can be both efficient and accurate for human eye gaze pattern investigation. For example, A. Cagatay Talay[4] used parallel genetic algorithm to detect human eyes in arbitrary images. Similarly, if we use genetic algorithms to model human eye gaze pattern, it can be faster and more accurate. Alternatively, if we would like to stick to RNN, we can further improve the algorithm. A. Mujika, F. Meier and A. Steger[5] proposed that it is possible to build block for Fast-Slow RNN by incorporating the strengths of both multiscale RNNs and deep transition RNNs, which can process different sequence length of datasets with different methods, such that the performance of the model can be improved.

Reference

[1]. Dingyun Z, B. Sumudu U. Mendis, Tom Gedeon, Akshay Asthana, and Roland Goecke: Hybrid Fuzzy Approach for Human Eye Gaze Pattern Recognition, The Australian National University.

[2]. T. Gedeon, Wattlecourses.anu.edu.au. [Online]. Available: https://wattlecourses.anu.edu.au/mod/resource/view.php?id=1833500..

[3]"Introduction to Recurrent Neural Network - GeeksforGeeks", GeeksforGeeks. [Online]. Available: https://www.geeksforgeeks.org/introduction-to-recurrent-neural-network/. [Accessed: 28- May- 2020].

[4]A. Cagatay Talay, "An Approach for Eye Detection Using Parallel Genetic Algorithm", Istanbul Technical University, Istanbul.

[5] A. Mujika, F. Meier and A. Steger, "Fast-Slow Recurrent Neural Networks", Department of Computer Science ETH, Zürich, Switzerland, 2017.