Comparison between Recurrent Neural Network and Simple Neural Network on Eye Gaze Recognition

Siqi Cui
Research School of Computer Science
Australian National University
u5717943@anu.edu.au

Abstract

Human eye gaze recognition is a crucial task in machine learning. Face perception and text reading are two of the most visual perceptual abilities for human eyes. In this paper, we constructed two neural networks to predict the human eye gaze recognition between text scanning and face scanning. One is a two-layer simple Neural Network, and another one is a Recurrent Neural Network. Comparing the results of these two neural networks, the Recurrent Neural Network performs better on test accuracy.

1 Introduction

The human eye's peculiar morphology reflects that the special role of eye gaze behaviour in daily life. Recognizing different eye gaze patterns is a prevalent machine learning task that various research projects have been carried out. Many approaches, such as deep learning and neural network, have been used to resolve this problem.
The neural network is a simulation of the human brain in order to achieve artificial intelligence-like machine learning technology. The neural network in the human brain is exceptionally complex, including an estimated 100 billion neurons and 1000 trillion synaptic interconnections. An artificial neural network is a widely parallel interconnected network composed of simple units with adaptability. Its organization can simulate the interactive response of the biological nervous system to real-world objects. It has the ability to learn from input data, explore the pattern, and predict the result.
Recurrent Neural network [6] is introduced in 1986 based on a work carried out by David Rumelhart. RNN is a generalization of feed-forward neural network that has an internal memory, which makes it performs well on time-series data. RNN is recurrent in nature because it performs the same operation for each data input, while the output of the current input depends on the previous one. It is copied and sent back into the recurrent network after the output has been produced. It considers the current input and the feedback it has learned from the previous input for making a decision[5].
In this paper, we will extend our previous work by applying a RNN and compare the results from both RNN and simple Neural Network to examine the performance on whether the pattern can be recognised.

2 Background

A time series is a series of indexed listed data points in time order. This type of data reflects the state or degree of change of a object or phenomenon over time. However, the data may not be time, it can be a text sequence.

The main point for this type of data is that there is a relationship between the current data and the previous one.
The figure below [3] shows a basic structure of a recurrent neuron. The recurrent neuron again has a connection between its output and its input, and therefore has three weights in this case. This third extra connection is called a feed-back connection, which enables the activation to flow in a loop around. They form a recurrent neural network when many feed forward and recurrent neurons are linked. The RNN has two advantages. The first one is that it can store information over time in form of activations. It is not memory-less. Another one is to learn sequential data.

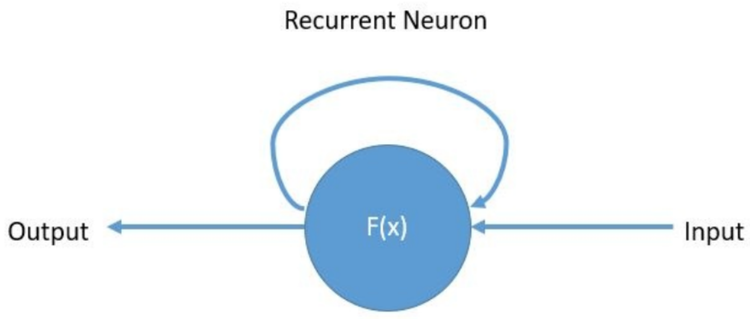


Figure 1: The structure of recurrent neuron.

The previous work explores how Bimodal Distribution Removal[7] performs on the neural network using eye gaze recognition data. However, the result shows that BDR doesn't perform well on real-world imbalanced data. It states that the BDR doesn't improve the accuracy with our dataset. This is caused by a lack of data points and imbalanced data. The recall for class 0 is much smaller than the one with class 1 in both networks. This is because of the imbalanced dataset. Some data which belonged to class 0 were categorised in class 1. This caused our low recall for class 0 and high recall for class 1 and then it leads to the overall accuracy reduction. Therefore, BDR is not used in this experiment.

3 Method

In this experiment, in order to examine the performance, we constructed two neural networks using PyTorch that one of them is a two-layer simple Neural Network and another one is a basic Recurrent Neural Network.

3.1 Data Processing

The data we used in this project is based on an eye gaze recognition research project[8]. There are four input features and one target and all data are divided into training and testing sets. Four input features represent the vertical distances between fixations: 1 & 2, 2 & 3, 3 & 4 and 4 & 5 set in the previous eye gaze recognition project.The target is either 0 or 1 indicated it is recognized or not. The data structure is shown in the table below.

Dataset	Target 0	Target 1	Total number of data points
Training	20	149	169
Testing	25	129	154

Table 1: Dataset structure

This table indicates that the data is strongly imbalanced that data categorised in class 1 is much more than data categorised in class 0. However, due to the lack of both training and testing data, we combined both datasets into one set, and divided them into 80% and 20% for training and testing purposes.
Furthermore, data normalisation has been done in this project. The goal of data normalisation is to translate the values of numeric columns in the dataset to a standard scale, without distorting differences in the ranges of values. However, since all input features of distances between fixations are in the same unit and scale, data normalisation is not really necessary in this case. After normalisation, all data will be range between 0 and 1.

3.2 Simple Neural Network

The simple neural network applied in the project is a two-layer neural network. This neural network contains 4 input features, 5 hidden neurons and 2 output classes. The hyper-parameters of the neural network include the number of hidden neurons, learning rate and the number of epochs. The hyper-parameters are justified and the accuracy results are shown in the table below.

	h = 5	h =10	h =50
lr =0.01	86.06%	86.06%	86.06%
lr 0.05	93.78%	95.34%	96.28%
lr 0.1	95.97%	96.28%	95.66%

Table 2: The accuracy of the simple neural network in different hyper-parameters.

The table illustrates that while the learning rate sets to 0.01, the accuracy is not changing because all outputs are categorised into class 1. That means the model has problems that data in class 0 are wrongly classified. Therefore, we set the learning rate as 0.05 and the hidden neurons as 5.
The neural network contains one sigmoid hidden layer. The sigmoid function is a non-linear function and ranges between 0 and 1. The properties of sigmoid function will lead in different "activation" probabilities for each output class. And we'll pick the one with the highest probability value of "activation." The architecture of the neural network is shown below in Figure 1.
The loss function used in the neural network is cross entropy loss function[1] in PyTorch. Cross entropy loss is commonly used for classification tasks. The formula for the binary cross entropy loss function is:

$$Loss = \sum_{i=1}^{n} y_i \log \hat{y_i} = -y \log \hat{y} - (1-y) \log 1 - \hat{y} \tag{1}$$

Where y is distribution for actual output values and network output distribution is ŷ.
Stochastic Gradient Descent (SGD) is used for the optimiser in this network. The reason to use SGD is that, in each iteration, it randomly selects an observation and updates the weight more frequently. Therefore, it performs fewer computations per iteration and is more able to manage noise, which helps it converge more easily.

3.3 Recurrent Neural Network

The RNN is implemented by PyTorch using its own RNN module. The network architecture is followed by an online tutorial [2]. The hyper-parameters of this neural network include the number of layers and the hidden layer dimension. The hyper-parameters are also justified and the accuracy results are shown in the table below.

	number of layers = 1	2	3	5
Hidden layer dimension = 5	91.2%	94.2%	94.2%	94.0%
Hidden layer dimension = 10	93.6%	93.2%	94.4%	95.0%
Hidden layer dimension = 20	94.4%	93.2%	94.0%	94.6%
Hidden layer dimension = 50	94.4%	94.8%	94.8%	94.8%
Hidden layer dimension = 100	94.0%	94.4%	95.4%	95.2%

Table 3: The accuracy of the recurrent neural network in different hyper-parameters.

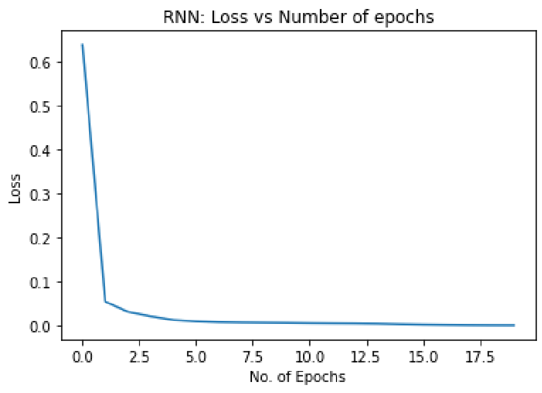
The table illustrates that the RNN performs better when hidden layer dimension of 100 with 3 layers. The learning rate is set to the same learning rate of 0.05 as the simpleNN one. However, the numbers are very similar since our lack of input features and data points. Both loss function and optimiser are using the same ones as the one in simple Neural Network. Loss function is the cross entropy loss and optimiser is the SGD optimiser.

4 Results and Discussion

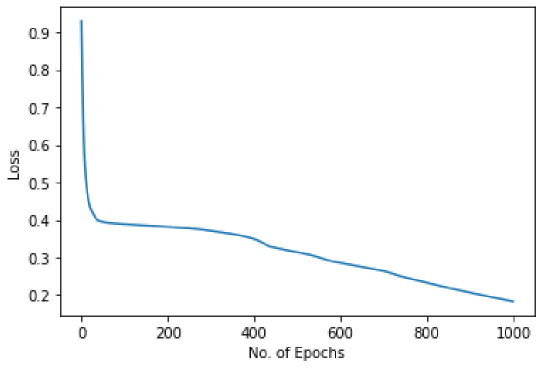
To effectively evaluate how two neural networks performs on our data, we construct some graphs and measurements to test the performance.

4.1 Recurrent Neural Network

Both loss graphs for RNN and simple neural network are shown below. Each unit for x_axis represents 50 epochs. It indicates that for both neural networks, the loss drops rapidly at the beginning and then it tended to be flat. However, the simple NN converges at 0.4 and RNN converges at 0.08.



(a) RNN: Loss versus number of epochs



(b) simpleNN : Loss versus number of epochs

Figure 2: Loss versus number of epochs for both neural networks

The graph below demonstrates the accuracy plot versus number of iterations in the RNN. The accuracy increases rapidly at the beginning and stays at the same for a while and drops a little and went back to 98% and then drops again. However, the difference is very small and this might be caused by the imbalanced data.
In the previous section, table 3 shows that for different hyper-parameters, the accuracy doesn't change much. This might be caused by lack of input features. In this experiment, only four input features are applied. In this case, it is not enough to improve the accuracy with this RNN.

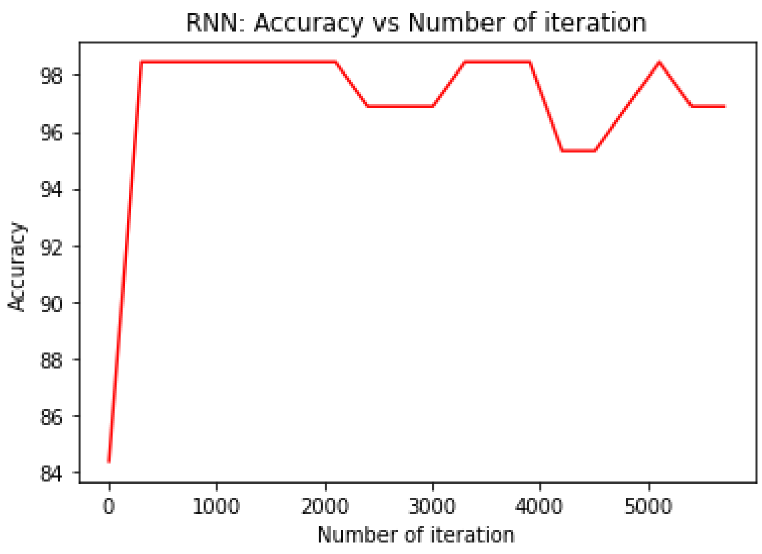


Figure 3: The accuracy versus number of iteration

4.2 Comparison with simple neural network

Some measurements have been used to evaluate the performance. The confusion matrix is determined for evaluation in this experiment. A confusion matrix is a table that is often used to define a classifier's performance on a set of test data for which the actual values are known. It enables the visualization of the performance of an algorithm. Accuracy and recall can be calculated using the confusion matrix. The confusion matrix, accuracy and recall are stated below.

	Class 1 Predicted	Class 2 Predicted
Class 1 Actual	TP	FN
Class 2 Actual	FP	TN

Table 4: The confusion matrix

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)} \tag{2}$$

$$Recall = \frac{(TP)}{(TP + FN)} \tag{3}$$

Accuracy and recall for simple neural networks are calculated and recorded in the table below. It shows that the results for RNN is better than simpleNN with almost 4%. Even though the dataset is imbalanced and our lack of input features, the RNN still performs better on sequential data than a simple Neural Network.

	Accuracy	Recall for class 0	Recall for class 1
simpleNN	91.63%	54.00%	98.16%

Table 5: The accuracy and recall table for simpleNN

4.3 Cross Validation

In order to predict how our model will perform on the datasets not used in training, we used cross validation to validate the model. Cross validation aims to improve on the basic methods of hold-out validation. K-Fold

Cross Validation is where a given set of data is divided into a K number of sections or folds where at some point, each fold is used as a test set. In this case, we set the K as 5. Therefore, the data set is split into 5 folds. The first fold is used to evaluate the model in the first iteration, and the remainder is used to train the model. In the second iteration, the second fold is used as the testing set, while the rest folds are used as the training set. This process is repeated until every fold of the 5 folds is used as the test set. Using cross validation can test every data in the dataset. The accuracy for both training and testing shows it is not over-fitting. In addition, using cross validation can handle an imbalanced dataset.

5 Conclusion and Future Work

This paper compares the performance for both simpleNN and Recurrent Neural Network using eye gaze recognition data. The results shows that the RNN has a slightly better performance. The accuracy increases 4% after using RNN.

In the future, we can improve the model by using LSTM[4] since LSTM is an advanced version of RNN. Genetic Algorithm was considered to use in this experiment. However, feature selection is not necessary when there is only 4 input features. In addition, the dataset is imbalanced. To handle this issue, we can resample the training set with over-sampling or under-sampling approaches.

References

[1] Loss functions — ml glossary documentation. https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html. (Accessed on 05/03/2020).

[2] Recurrent neural network with pytorch | kaggle. https://www.kaggle.com/kanncaa1/recurrent-neural-network-with-pytorch/notebook?scriptVersionId=31582568. (Accessed on 05/20/2020).

[3] Recurrent neural networks - combination of rnn and cnn - convolutional neural networks for image and video processing - tum wiki. https://wiki.tum.de/display/lfdv/Recurrent+Neural+Networks+-+Combination+of+RNN+and+CNN, 2017. (Accessed on 05/25/2020).

[4] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[5] Aditi Mittal. Understanding rnn and lstm - towards data science. https://towardsdatascience.com/understanding-rnn-and-lstm-f7cdf6dfc14e, Oct 2019. (Accessed on 05/24/2020).

[6] Hinton G. & Williams R. Rumelhart, D. Learning representations by back-propagating errors | nature. Oct 1986. (Accessed on 05/29/2020).

[7] P. Slade and T. D. Gedeon. Bimodal distribution removal. In José Mira, Joan Cabestany, and Alberto Prieto, editors, *New Trends in Neural Computation*, pages 249–254, Berlin, Heidelberg, 1993. Springer Berlin Heidelberg.

[8] Dingyun Zhu, B. Sumudu U. Mendis, Tom Gedeon, Akshay Asthana, and Roland Goecke. A hybrid fuzzy approach for human eye gaze pattern recognition. In Mario Köppen, Nikola Kasabov, and George Coghill, editors, *Advances in Neuro-Information Processing*, pages 655–662, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.