An Empirical Study of Applying LDA and GA Feature Selection on Feed Forward Neural Network and Cascade Network (Casper) for Multi-Classification Task with Small Depression Dataset

Xiangyi Luo
Research School of Computer Science
The Australian National University
Canberra, Australia
u6162693@anu.edu.au

Abstract. The previous research manifests that physiological responses observing individuals with depression can assist to estimate depression levels, which is more accurate than the subjective prediction by observers and is helpful to psychological diagnosis [1]. Feed Forward Neural Network (FFNN), a classic and powerful network, is utilized widely in the multi-classification problem while the cascade network is less popular. Cascade-forward networks are similar to FFNN, except that each hidden neuron is connected from the input and all pre-installed hidden neurons. One fundamental example is the Cascade Correlation (Cascor) [2]. Cascade Network Algorithm Employing Progressive RPROP (Casper) has a similar network structure with Cascor but was shown to produce more compact and generalizes better than Cascade Correlation (Cascor) [3]. As the depression dataset has few records (192) and many features (85), Linear Discriminant Analysis (LDA) and genetic algorithm (GA) were employed for feature selection to overcome overfitting issues. Under settings in this paper, the experimental results show that LDA and GA improve the overall accuracy of FFNN and Casper. FFNN+LDA gets better performance than Casper+LDA while FFNN+GA and Casper+GA get semblable improvement. Among the four combinations of feature selection techniques and model, FFNN+LDA has the best performance.

Keywords: Forward Feeding neural network, Cascade Network, Latent Dirichlet Allocation, Genetic Algorithm for Feature Selection, Small datasets, Depression Detection, Physiological Signals

1 Introduction

1.1 Background

Clinical depression can cause serious consequences like commit suicide, normally, it is difficult for people without any professional training to recognize others' depression precisely. To diagnose depression levels, doctors can use several tests such as physical and psychological exams. However, subjective diagnose might not be sufficiently accurate. An experiment involving 12 individuals shows that the accuracy of subjective prediction of depression level is only 27% [1]. A novel idea suggests taking advantage of observers' physiological responses of video to assist in predicting depression levels of video [1]. The experiment requires participants entering the laboratory room, physiological signals are collected when they watch videos [1]. As the number of participants is limited and the signals are time series data, the dataset was small and with relatively high dimensions, the details of the dataset will be demonstrated in session 1.2.

The study is to compare the standard forward feeding neural network (FFNN) to the variation of cascade network (Casper) with two feature selection techniques given a dataset with few records (192) and a relatively large dimension (85). To relieve the overfitting problem, Linear Discriminant Analysis (LDA) and genetic algorithm (GA) are employed. The performance of networks will be measured according to one-vs-all precision, recall, F1 score, and the overall accuracy.

1.2 Dataset

The depression dataset is from an experiment done by Zhu et al [1]. Physiological Signals were collected from participants when they were watching 16 videos of a person having different depression levels. Three signals are Galvanic Skin Response (GSR), Skin Temperature (ST), and Pupillary Dilation (PD) [1]. The statistical measures such as minimum, maximum, mean, variance, numbers and amplitudes of peak occurrences, and so on were extracted from the normalized time-series signal data. As a result, the dataset contains 23 features of GSR, 23 features of ST, and 39 features of PD. The ground truth depression levels contain four categories: no depression, mild depression, moderate depression, and severe depression. Four depression levels were labelled as numbers: 0, 1, 2, 3.

2 Method

FFNN and Casper were implemented from scratch by PyTorch and other common packages. FFNN contains one hidden layer. Casper has one hidden neuron initially and the new hidden neuron is added once the current whole network is fully trained.

In paper [1], putting all features in an FFNN gives an overall accuracy of 88% and the use of GA for feature selection improves the accuracy to 92%. For comparison, all features will be used for FFNN and Casper in this empirical study. The feature selection techniques are discussed in session 2.1. The details of network structures and hyperparameter settings for FFNN and Casper are in session 2.2 and 2.3 separately. A short comparison of the two networks is in session 2.4.

2.1 Data Preparation and Feature Selection

Firstly, all columns are standardized by a robust scaler as it is robust to outliers. The control group is taking all features into FFNN and Casper without feature selection. Two experimental groups are applying LDA and GA for feature selection.

LDA - LDA is a popular feature extraction technique for classification tasks given labelled training data. Since the performance of LDA would be affected by collinear variables, principal component analysis (PCA) was employed to project data into a 10-dimensional subspace with the largest eigenvalues to avoid collinear variables. Then, as the depression levels separate into four categories, multiclass LDA will find a three-dimensional subspace to contain all class variability [4].

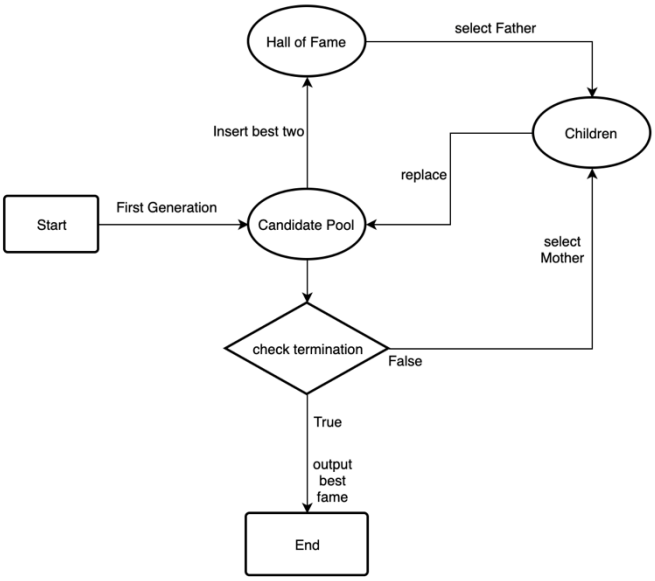

Fig. 1. The Genetic Algorithm flow chart.

GA - A genetic algorithm is designed to select a feature subset. The flow chart is displayed in fig 1.

Population – There are 10 individuals for each generation.

Chromosome – A potential subset is considered as an individual in GA and represented by a binary string of length 85. Each bit indicates the occurrence of the corresponding feature. For instance, if the first character of the binary string is 0, then the first feature is eliminated, otherwise, the first feature is retained.

Hall of fame - For each generation, the best two individuals will be inserted in the hall of fame. The capacity of the hall is 10 and if the hall is full, bad individuals will be eliminated.

Selection – Two parents are selected to generate children. The father is selected from the hall of fame on equal probability. The mother is selected based on linear rank in the current candidate pool, which allows more diversity. The selecting weight of each individual is computed by equation **(1)**. The variable pos is the rank of an individual in the population sorted by fitness value in ascending order. The selection pressure SP should in the range [1, 2] and is set to 1.5 in this paper.

$$Weight(pos) = (2 - SP) + 2 \times (SP - 1) \times \frac{pos-1}{pop\ size-1} \quad \textbf{(1)}$$

Crossover – The crossover operator is applied to two parents according to the crossover rate, $P_c = 0.8$. A randomly generated mask is used to determined bits of a chromosome to crossover.

Mutation – The mutation operation flips the bit and mutation rate p_m was dynamically determined by equation **(2)** where t is the generation counter. It is relatively large at the beginning and decreases exponentially with the generation number [5].

$$p_m(t) = \frac{1}{140} + \frac{0.11375}{2^t} \quad \textbf{(2)}$$

Fitness Function – For a given features subset, the overall accuracy of 12 models in one leave-one-participant validation is the fitness value. The leave-one-participant validation will be explained in the result and discussion session.

When to stop – For each generation, record the average fitness value of the hall of fame. If three consecutive generations have average fame fitness within difference 0.001, the algorithm terminates.

Output – When the algorithm terminates, the best individual in the hall of fame is the best subset of features selected by GA.

2.2 Feed Forward Network

FFNN was simple. The three-layer network is fully connected. The sigmoid function is used as an activation function for hidden-layer neurons. The softmax function is employed for output layer neurons and the cross-entropy is then computed. Adam optimizer is used for weight updating. Based on the experiment, the good choice of learning rate is 0.01 and the maximum epoch was set to 3000. For every 50 epochs, the loss decrement will be calculated for convergence checking. If the loss decrement is less than 1% of previous loss, the training is considered converged.

2.3 Casper Network

The technique for the Casper model is more complicated. According to [], initially, the Casper network involved only one hidden neuron. The inputs for this hidden neuron are coming from all first-layer neurons. Also, all first-layer neurons and this hidden neuron are linked to each output neuron. Then, applying progressive resilient backpropagation (RPROP) to update weights. For the first train, the initial step for all weights w_{ij} is 0.2. Then, a new hidden neuron will be installed once the whole network is considered fully trained [3]. The new hidden neuron will take the output of all input-layer neurons and installed hidden neurons as input and pass its output to the final output layer.

To measure whether the current network was well trained, we need to check the loss drop for every period. The period is computed in formula **(3)** where the P is user-defined constant and N is the number of hidden neurons installed [3]. For this depression level classification task, the constant P was set to 0.5.

$$time\ period = 15 + P \times N \quad \textbf{(3)}$$

The backpropagation process was a core part of the Casper algorithm. The Progressive RPROP combined the concepts of Simulated Annealing (SA) and RPROP. The SA term was used for weight decay, introducing penalty according to the square of weight and reducing model complexity. The error gradient function was shown as equation **(4)** where $HEpoch$ is the number of epochs passed since the installation of the last hidden neuron. Based on [3] and [5], the constant k is problem dependent and for this task k was set to 1e-4 to achieve convergence.

$$\frac{\delta E}{\delta w_{ij}} = \frac{\delta E}{\delta w_{ij}} - k \times sign(w_{ij}) \times {w_{ij}}^2 \times 2^{-0.01 * HEpoch} \quad \textbf{(4)}$$

Unlike Cascor network freezing all previous weights after adding a hidden neuron, the whole Casper network is separated into three regions L1, L2, and L3 as shown in **Fig.2**. Every time a new hidden neuron is added, the initial learning rates should be reset [3]. According to [3], usually, the initial learning rate should be L1 >> L2 > L3. As mentioned in paper [3], the initial step for L1, L2, and L3 were found to be problem independent, and hence I used the initial learning rate setting as the same as the technique paper, which is L1 = 0.2, L2 = 0.005 and L3 = 0.001.

The activation function in hidden neurons is the hyperbolic tangent function and the output activation function is softmax function. The loss is cross-entropy as well.

2.4 Technique comparison

FFNN and Casper are similar as they both forward from the input layer to the output layer and update weights by backpropagation.

There are still some differences. For the feed-forward network, there were no links between the input layer and output layer, the inputs should pass through all hidden layers and arrive at the output layer. For the Casper network, each hidden neuron is linked to all input neurons, all installed hidden neurons, and all output neurons. Normally, the number of weights in the Casper network is greater than weights in a fully-connect feed-forward network.

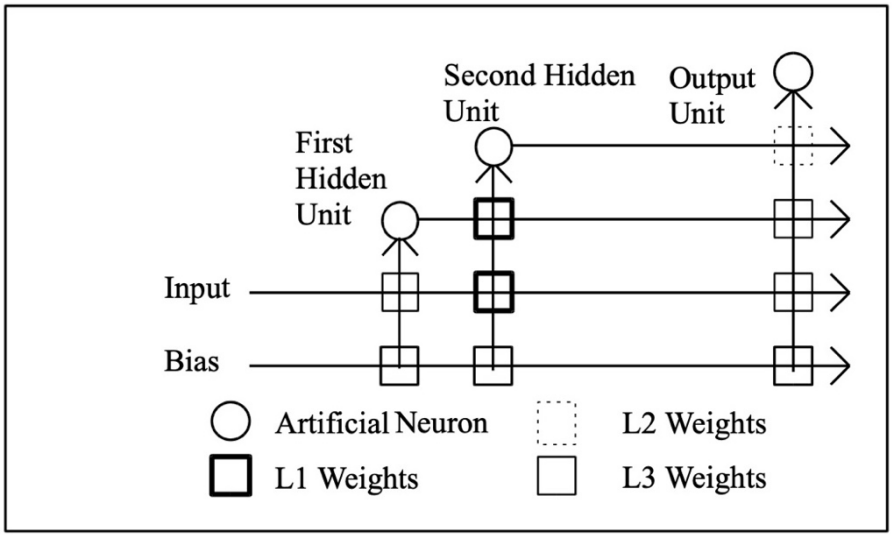

Fig. 2. The Casper architecture, the second hidden unit just installed [3].

3 Results and Discussion

As k-fold cross-validation is not suitable for the depression dataset, the leave-one-participant-out mentioned in [1] does similar work with k-fold cross-validation. The leave-one-participant-out ensures that using data from a participant untouched by the model during the training process for testing. For each iteration, all records about one specific participant are testing data while the records of other participants are training data. Thus, each evaluation process trains 12 different models and overall accuracy is the measure of performance of the model.

The **control group** is taking all features into FFNN and Casper without feature selection. **Two experimental groups** are using LDA and GA for feature selection. For testing and comparison, networks are trained with 5 hidden units. In this session, the results of the control group and experimental groups will be discussed.

3.1 All Features

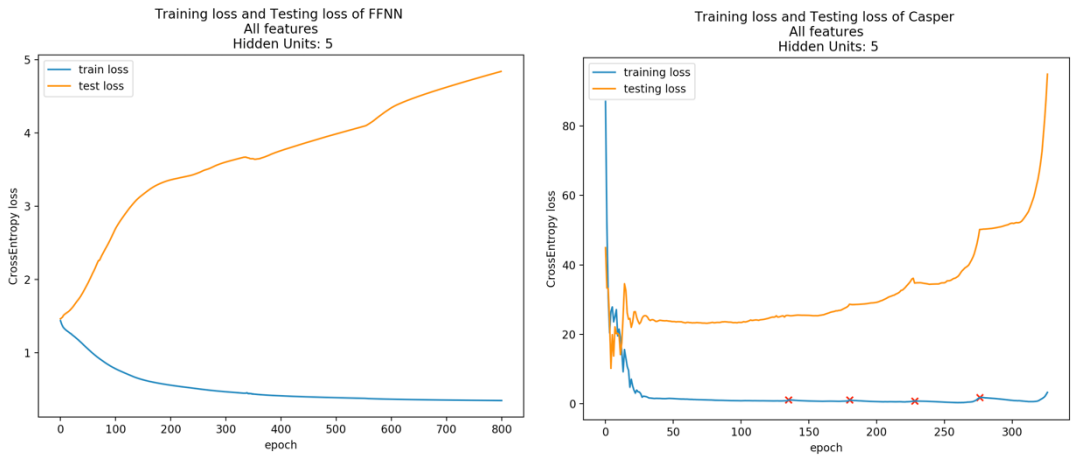

Fig. 3. Taking all features as input for FFNN and Casper network, displaying training loss (blue curve), and corresponding testing loss (orange curve) during training. The red cross in the right diagram indicates the installation of a new hidden neuron.

It is not surprising that using all features directly causes the overfitting problem. **Fig.3** shows the training loss and testing loss in one training process while setting 5 hidden units in both FFNN and Casper.

3.2 All Features + LDA

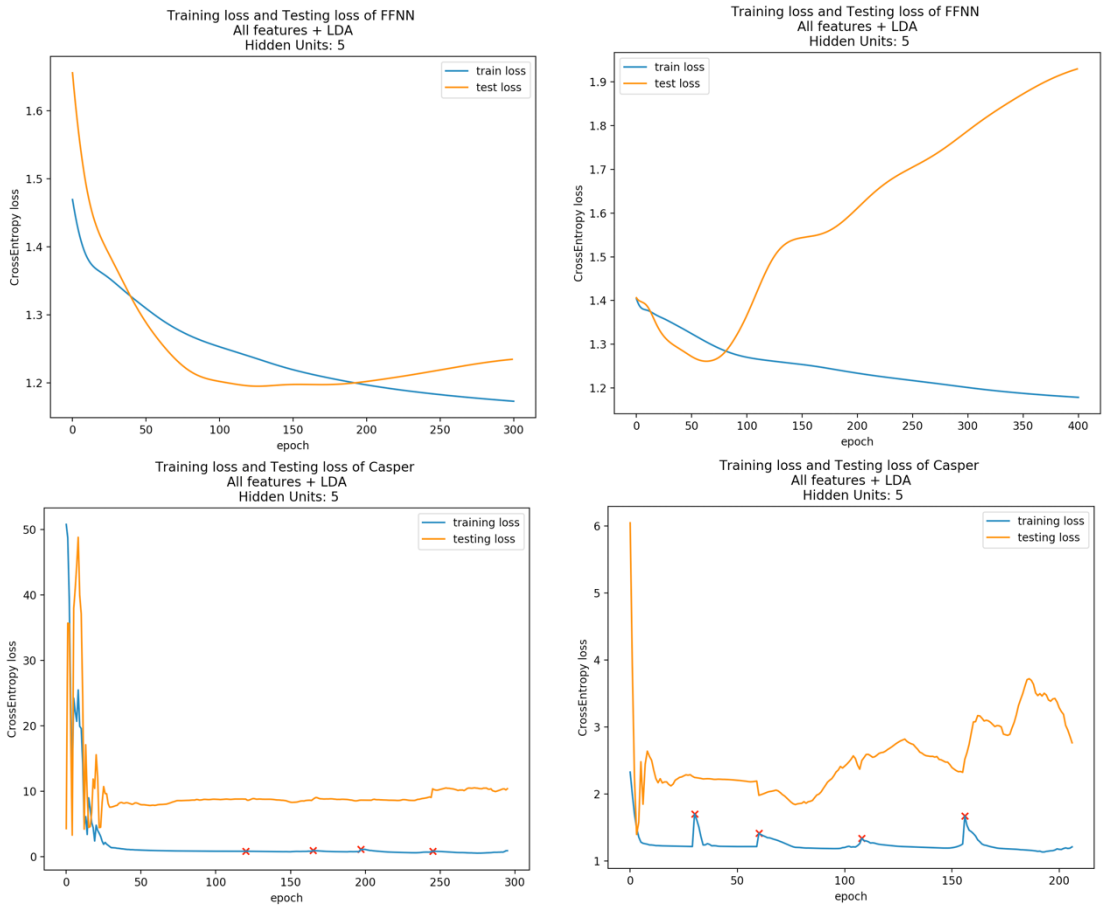

Fig. 4. Applying LDA on all features as input for FFNN and Casper network, displaying training loss (blue curve), and corresponding testing loss (orange curve) during training. The red cross in the second row's diagrams indicates the installation of a new hidden neuron. The left column shows examples that overfitting issues are deduced while the right column is not.

In **Fig.4**, training loss and testing loss when applying LDA for feature extraction for both FFNN and Casper with 5 hidden neurons are demonstrated. In one leave-one-participant-out validation including 12 models, the overfitting issues in more than half models are deduced (as shown in the left column in **Fig.4**) while a few models still have overfitting problems (as shown in the right column in **Fig.4**).

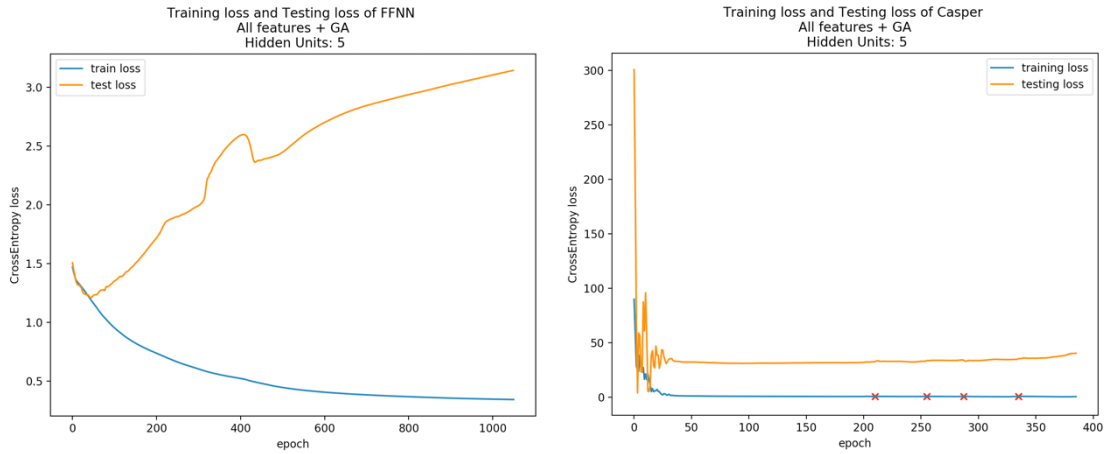

Fig. 5. Applying GA on all features as input for FFNN and Casper network, displaying training loss (blue curve), and corresponding testing loss (orange curve) during training.

3.3 All Features + GA

As shown in **Fig 5**, the overfitting problem in FFNN is not deduced by GA but the testing loss is less than FFNN without GA. For Casper, the overfitting problem of half of 12 models is relieved with the assist of GA.

3.4 Performance Comparison and Discussion

To measure performance, different combinations of feature selection techniques and networks were run. I ran 5 times of leave-one-participant-out validation and computed average overall accuracy for the control group and experiment group with LDA, which covers the number of hidden neurons in the range [1, 19]. As the GA feature selection process is very time-consuming, I only run the experimental group with GA once for hidden units in the range [1, 10]. Also, the training time is recorded as the time cost is also a measure of model performance.

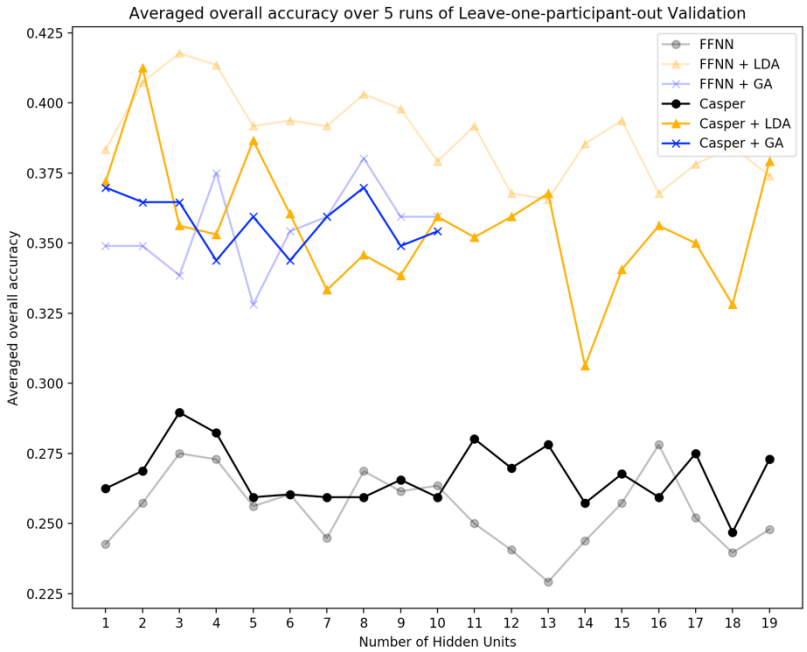

Fig. 6. Accuracy for the control group (black color) and the experimental groups (blue and orange color) with a different number of hidden units.

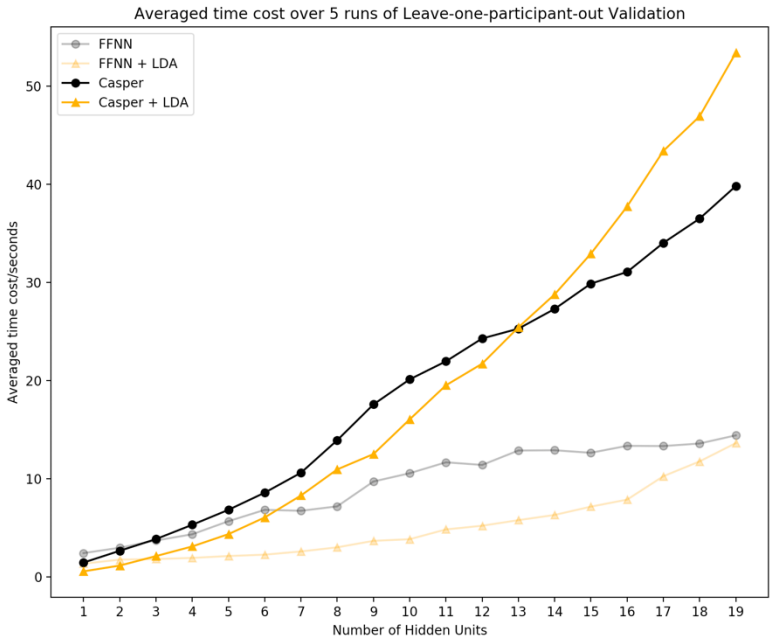

Fig. 7. Time cost for the control group (black color) and the experimental groups with LDA (orange color) with a different number of hidden units.

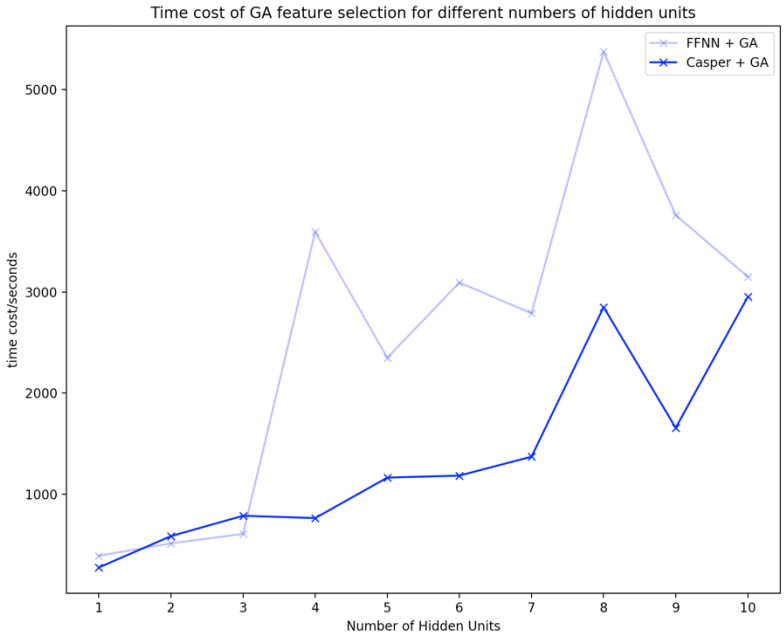

Fig. 8. Time cost for the experimental groups with GA (blue color) with a different number of hidden units.

As shown in **Fig. 6**, for the control group, Casper does slightly better than FFNN. With LDA, performances of FFNN and Casper are improved. FFNN+LDA does better and has more stable improvement than Casper+LDA. GA feature selection also improves FFNN and Casper performance but not that much as FFNN+LDA.

As shown in **Fig. 7**, the time cost increases as the number of hidden units increases. Generally, FFNN requires less time cost than Casper. **Fig. 8** indicates that the time costs of GA are very large. The time costs of FFNN+GA and FFNN+Casper for hidden units from 1 to 3 are similar. For hidden units greater than 3, they take more time for FFNN+GA to converge than Casper+GA.

4 Conclusion and Future Work

In this empirical study, as both feed-forward network and Casper network were built from scratch, the implementation differences between the two models were experienced clearly. As the network structure was dynamically changed, the implementation of Casper was more complex. From session 3 we could notice that the performance differences were not obvious between two models without feature selection. This might because part of the details discussed in technique paper was not implemented such as using hyperbolic arc tangent function as an error function for overcoming the 'flat spot' problem.

In conclusion, both LDA and GA improve performances of FFNN and Casper. Based on the parameter settings in this paper, FFNN+LDA performs better than Casper+LDA. FFNN+GA and Casper+GA have similar performances. LDA is easy to use but not very flexible since it needs a labelled training dataset and the dimension of the subspace is limited. GA is more flexible and more intelligent but needs much more time to converge and needs more experience-based adjustment for the evolution algorithm. In this paper, due to time and hardware limitations, the population size is small, which causes that the exploration of the search space is small.

In the future, the details of the Casper model will be completed. Besides, the different population size, selection pressure, and evolutionary algorithms in GA will be tested to see if any further improvement of models happens.

References

1. Zhu X, Gedeon T, Caldwell S, Jones R. Detecting Emotional Reactions To Videos Of Depression.
2. Fahlman S, Lebiere C. The cascade-correlation learning architecture. Advances in Neural Information Processing Systems 2. 1990.
3. Treadgold N, Gedeon T. A cascade network algorithm employing Progressive RPROP. Biological and Artificial Computation: From Neuroscience to Technology. 1997:733-742. doi:10.1007/bfb0032532
4. Linear discriminant analysis. En.wikipedia.org. https://en.wikipedia.org/wiki/Linear_discriminant_analysis. Published 2020. Accessed May 31, 2020.
5. Engelbrecht A. Computational Intelligence: An Introduction. 2nd ed.; 2017: Chapter 9.