Comparing Casper Network with standard Feed Forward network with various training strategies on music_effect data set

Luyang Zhou
Research School of Computer Science, Australian National University,
Canberra Australia
U6872319@anu.edu.au

Abstract: Casper is one of the constructive networks which has some advantages over standard neural network architecture. In this paper, we explored and compared the differences between the two architectures based on different training strategies. We found that Casper slightly outperformed standard NN by 4.5% in gradient-based training and 7% in genetic algorithm training. Moreover, during both training and testing processes, Casper network converged with less oscillation compare to standard NN due to different natures of Rprop and RMSprop. However, none of the two networks is better than the result shown in the original paper since we only used 4 out of 25 most important features mentioned in the original paper. Therefore, future work may be done to include more crucial features or acquire more data.

Keywords: Casper, NN, Rprop, effect EEG, genetic algorithm

1 Introduction

1.1 Motivation

The standard feed forward network has been proved to be a powerful tool in comparison with traditional machine learning algorithms in many classification and regression problems. Feed forward neural network automatically extract features from imported training samples, and uses activation functions to model non-linearity. Therefore, it can mimic any kind of patterns theoretically. However, apart from various advantages, researchers and practitioners also discovered many issues and drawbacks for standard neural networks, among which, the difficulties in tuning hyper-parameters to decide the topology of the model is considered a common one. Moreover, gradient descent with back propagation, the most popular algorithm in training, often suffers from slow and computationally expensive learning process. Although sometimes it can be mitigated by other algorithms like Adam and RMSprop, if the network is too deep, it is still an intricate problem.

The Cascade Correlation algorithm (Cascor), proposed by Fahlman and Lebiere, is considered a fairly effective solution to major issues for feed forward network in some complicated classification problems[1]. It is a constructive algorithm which starts with a minimum network contains only an input layer connected to an output layer. After having trained the minimum network, one hidden neuron is added in the middle of the network immediately before the output layer. This hidden neuron receive connections from input neurons. When adding a new hidden unit in Cascor, all weights in the previous network are frozen, then all the incoming weights are trained to optimize the covariance between the hidden units value and the residual output error. After the hidden unit is added to the network, we train the output layer to minimize the residual error of the network[2]. The above process will be done repeatedly until the whole network reaches the desired result. Since this network architecture trains many separate 'mini networks' rather than having to back propagating the whole feed forward network, the training process would be far more efficient especially for very deep networks[1].

However, Kwok and Yeung suggested (1993) in their work that Cascor is not flawless[2]. Firstly, they pointed out that the "frozen weights" step described above can generate a very deep network due to inefficient feature detection for early added hidden units. Besides, it does not generalize very well in some classification problems[3]. Therefore, a modified version of Cascor, namely, "Casper network with pregressive RPROP" was invented by N.K. Treadgold and T.D. Gedeon[4]. The overall ideas in Cascor and Casper are very much alike, except for that in Casper, there is no need to freeze other weights when adding hidden units. Additionally, instead of using covariance as the measure, different learning rates are applied when adding hidden units region-wise[4]. According to their work, Casper converge slower than Cascor in training, but generates shallower network with a slightly higher testing accuracy. This report will compare the performance between Casper and standard feed forward network in a music genre classification problem based on a subset of music effect EEG data set. Except for standard gradient descent, genetic algorithm will also be applied for training to explore the architectural advantages for Casper network.

1.2 Data set

In the experiment, we will be using a subset of "Effects of Music on Brainwave Patterns"[5]. The data is collected as electroencephalogram (EEG) from 13 male and 11 female students (24 students in total) studying at Australian National University. Originally, the data contains 364 features extracted from 14 channels per song and 26 statistical features per channel. By applying feature selection techniques, the authors finally screened out the most important 25 features, from which only 5 of them are in channel F7 (Frontal Lobe). However, the data we received contains 25 features only in channel F7 with music genre labels for all data samples, which is different from the original version.[5]

2 Method

2.1 Data Preprocessing

To start the experiment, we performed data preprocessing since after inspection, the data are not yet ready to feed directly into networks.

The data we received have 576 samples, 26 features and 1 label represents the music genre for each single sample. We deleted the first column which is the candidates' number since it apparently has no effect for our task. We also noticed the labels fall in the range of (1,3], which is not the required range if we use cross entropy loss in pytorch, therefore, we subtracted one from each label to make sure labels range from (0,2]. Then, examples and labels were separated by extracting the last label column from the original dataset. We now have two Dataframes contain pure examples and labels respectively.

After having checked that the labels are perfectly balanced in the dataset(with exactly 192 examples for each labele), we drew a column wise box plot to see whether the data have significant outliers. The box plot is show in the graph below.

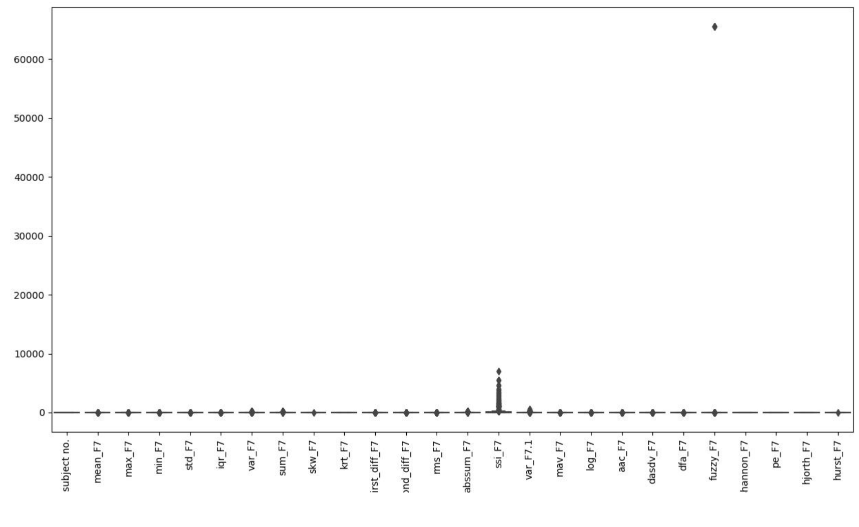

Fig 1. Column-wise box plot for the data

Obviously, a significant outlier is detected in column fuzzy_F7 with value 65535. We replace this value by the median of the column. Next, we perform candidate-wise Min-Max standardization rather than Z-score normalization, as some candidates' EEG data are too much higher than others, e.g 3rd and 8th candidate. Then we split the data samples and labels into three portions, 80% for training, and 10% each for validation and testing. Notice that data has been shuffled and stratified during splitting. Now our data are ready to be feed into neural networks.

2.2 Standard feed forward neural network

It is straightforward to implement a standard feed forward neural network in Pytorch. The model structure is shown below.

- Start the network with an 25-units input layers.
- There is a batch normalization layer before the first hidden layer.
- The first hidden layer has 8 hidden units, followed by a relu activation function.
- The second hidden layer has 4 hidden units, followed by another relu activation function.
- The output layer has 3 units, and it is wrapped by a softmax activation function.

Similar to most multi-classification problems, here we use cross entropy as the loss function and RMSprop optimizer with default settings. Besides, other hyper-parameter settings for training are listed below.

- Learning rate is set to be 0.005
- We train the network for 1000 epochs
- In order to be able to compare with Casper with RPROP optimizer, which is incompatible with mini-batch, here we use full batch for training.

To measure the performance of the network, we use accuracy and confusion matrix, by which it is obvious to see the details of the testing results.

2.3 Implement a Casper network

The Casper network is more complicated to build since it requires customization. The overall architecture can be illustrated in the following graph.

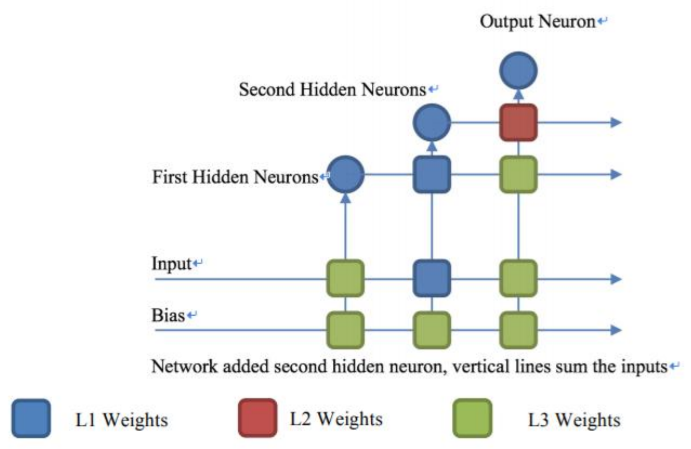

Fig 2. Structure of Casper network and different learning rate based on position of the neurons[6], note that in our task,the output layer should have three units.

To begin, we initiate a Casper class as a child class of torch.nn.Module. We maintain several attributes in the model class, a list of linear layers to represent connections among input and hidden layers(units), a relu activation function and the output activation function softmax. Every time when we add a hidden unit, we instantiate or modify an linear output layer to reflect the new incoming weight from newly added hidden unit. Then, we connect the hidden unit with all input, output units, as well as previously added hidden units. At last, we train the existing network as a whole using RPROP. In each forward pass, we concatenate the data flow from the previous layer with one more hidden unit which we just added, and wrap the intermediate layers by a relu activation functions and add a Batch Normalization layer to fix the data distribution deviation. We can see that unlike Cascor network, in Casper network, each newly added hidden unit is directly connected to all other existing units. This allows every hidden units in the network to be fully trained instead of freezing the previous ones.

In Casper, we assign different learning rate according to Figure 1 each training cycle. Usually L1 is much higher than both L2 and L3, and L2 is slightly higher than L3. The reason behind this is very much like "training two smaller networks" idea in Cascor. A higher value of L1 helps the new hidden neuron to learn remaining network error, and L2 allows the neuron to reduce the network error[4]. Since other weights are all equal to L3, which is smaller than both L1 and L2, the learning process for the new hidden neuron is not interfere too much by other weights.

In order to set up dynamic learning rate, first, before training, we set learning rate for all weights to be L3. Then, for each time we add a new hidden unit, we update the layer-wise learning rate for Rprop optimizer as follows,

1. Set the learning rates for all incoming weights/bias of the new hidden unit(immediate previous layer) to L1=0.005
2. Set the incoming weights/bias from new hidden unit to output layer to L2=0.0002.
3. Leave other weights in the network to L3=0.0001.

Every time after we added a new hidden units, we train the entire network for 150 epochs. The maximum hidden units we add is 8.

Note that whether updating learning rate for bias or not does not matter too much, as the bias matrix usually contain much less parameters than weight matrix. Therefore, has only minor effects for learning.

2.4 Training with Genetic Algorithm

For the purpose of separating the reasons why the performance of standard feed forward network is different from Casper network, we used genetic algorithm to train the two networks. Since genetic algorithm is purely stochastic and based on populations, which spares us from using optimizers. Therefore, we can investigate whether the architecture of Casper network itself has some advantages over standard feed forward network. We will use pyGAD to do the training. Admittedly, training with EA is very expensive, but as our data set is not too large, the training time is still acceptable. The hyper-parameters for training are shown below.

- We use 80% data for training and 10% each for validation and testing.
- Steady state selection method will be used for parent selection to replace those low-performance models with new, and better offsprings.
- We use "two-points" crossover method.
- Mutation will happen randomly with equal probability (12%) to every single chromosome.
- Number of generations is set to be 500.
- We keep all the parents to the next generation.
- We will have 50 solutions in the population. This hyper-parameter cannot be set too large as it significantly increase the computation time.
- The fitness function is the inverse of cross entropy loss which we need to maximize.

Note that we will keep all model hyper-parameters for our two models same as before.

3 Results and Discussion

3.1 Compare results for feed forward network and Casper network

We now explore how Casper performs compare to standard feed forward network.

We run the two networks with hyper-parameter settings defined above under the same random seed, the results are shown in the following table.

Feed forward	train_acc	test_acc		**Casper**	train_acc	test_acc
1	80.43%	43.10%		1	55.43%	48.28%
2	78.91%	37.93%		2	58.04%	44.83%
3	76.74%	39.66%		3	58.48%	48.28%
4	82.61%	44.83%		4	54.57%	44.83%
5	78.70%	44.83%		5	54.78%	46.55%
Mean	79.48%	42.07%		**Mean**	56.26%	46.55%
std	3.02%	2.18%		**std**	1.86%	1.99%

Fig 3. training and testing results for standard feed forward network and Casper network on music EEG data

We see that Casper network consistently outperforms standard NN by about 4.5% in test accuracy. Besides, the results for Casper are less volatile in both training and test accuracy.

Looking at the result of the feed forward network, significant over-fitting is detected due to an about 42% testing accuracy but nearly 80% training accuracy. We tried batch normalization, L2 regularization and trained the network for less epochs, but none of them improved the test accuracy. In addition, from the perspective of the dataset, we tried to drop some features, but did not work. On the contrary, Casper network is less over-fitted. This is because we trained the entire Casper network for less epochs every time we added a new hidden unit.

We can investigate the details on over-fitting from how training loss and validation loss evolve during training run.

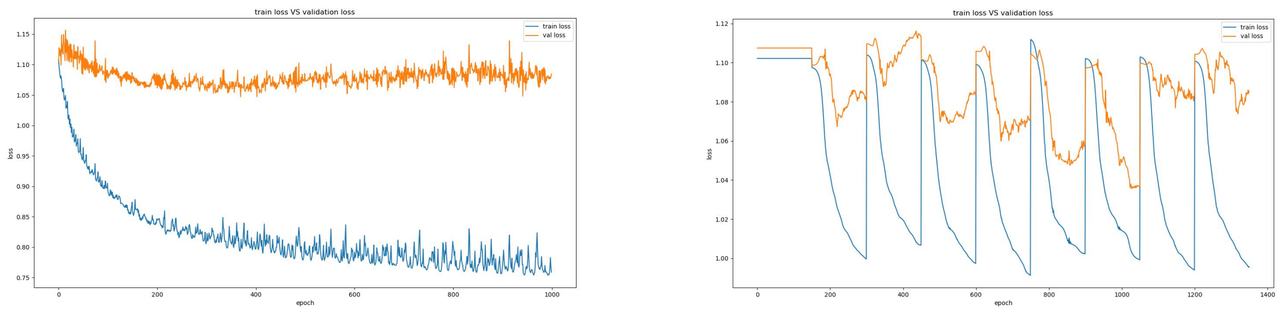

Fig 4. training loss vs. Validation loss for feed forward network(left) and Casper network

We can see from the left graph that for feed forward network, validation loss first decreased for about 200 epoch and then stay on that level until the end of the training. However, although there is oscillation, the trend of training loss is always downwards. As the gap between training loss and validation loss kept expanding, the over-fitting started at the early stage of the training.

For Casper network, the situation is better since we have less training epochs. The validation loss first dropped with the training loss for about 100 epochs and then bounced back.

We also noticed that the training loss decreased much smoother and quicker due to the nature of RPROP optimizer. In RPROP, if we find the right direction for gradient, the step size will keep increasing multiplicatively, which accelerates the learning process. But RMSprop in feed forward network is based on the concept of exponential weighted moving average, which averages out the step sizes in training.

The other reason why Casper performed better and less over-fitted might be the network architecture. In Casper, each hidden neuron has connections to all other neurons. Whereas in feed forward network, hidden neurons do not connect with other hidden neurons in the same layer, therefore, capture less information. This is in line with the fact that we only used up to 8 hidden neurons in Casper network and obtained a better performance than a feed forward network with 12 hidden neurons.

3.2 The effect of early stopping

In order to find the “turning point” for validation loss in both two networks, we implemented them again with early stopping with all settings unchanged. In addition, the “patient” is set to 10 for our two models, which means if the validation loss does not decrease for 10 epochs, the training session will be terminated.

As we expected, after running many times, for standard NN, the training usually stopped before 30 epochs, with very random test accuracy roughly ranging from 30% to 50%, meaning very limited knowledge has been learned. On the other hand, the results for Casper network is shown below.

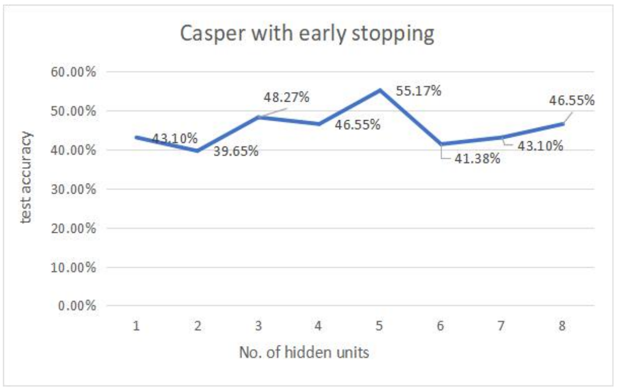

Fig 5. One example from many training runs for Casper network with early stopping

We can see Casper network still performed better than both standard NN with early stopping, and Casper network without early stopping. However, since the result is still significant lower than the original paper, we believe the volume of data set we used are not enough and well distributed to detect clear pattern. This can be proved by running a 10-fold cross validation training, in which ten resulting test accuracy are very different

In fact, the dataset we use here is only a subset, which contains only 4 out of 25 most important features across 14 channels mentioned in the original paper[5]. They are far from enough to represent the characteristics for 3 types of music genre. If we can collect more data or meaningful features, clearer pattern may be discovered.

3.2 Training with genetic algorithm

To prove the advantages of the architecture of Casper network, we used genetic algorithm to train our two networks. Again, we keep all network hyper-parameters as before, and set hyper-parameters for genetic algorithm as discussed in the previous section. However, we do not use any optimizers since our algorithm is purely based on populations.

As it takes too long to run one training session for our Casper network, we only trained it once. The result of 5-times training for standard NN and 1-time training for Casper network are shown in the following tables.

no. train	train accuracy	test accuracy
1	53. 48%	43. 10%
2	52. 61%	41. 38%
3	53. 48%	41. 38%
4	54. 13%	41. 38%
5	53. 91%	36. 21%

accuracy after no. hidden units added	train accuracy	test accuracy
1	54. 56%	44. 83%
2	52. 39%	50%
3	52. 61%	48. 28%
4	55. 87%	48. 28%
5	52. 61%	46. 55%
6	50. 87%	46. 55%
7	51. 96%	41. 38%
8	51. 96%	37. 93%

Fig 6. Training results with genetic algorithm for standard NN(left) and Casper network

We can see Casper network is still better than standard NN. The training and test accuracy of Casper network reached a high level after second, third and fourth hidden units were added. Therefore, we can conclude that Casper network is more powerful on our data set as it can achieve better performance even with simpler network structure and fewer parameters.

4 Conclusion and future work

In this paper, the performance between a standard feed forward network, and a Casper network are compared based on music effect EEG dataset. Although Casper network performed better than standard NN in various training methods, the results are still significantly lower than which in the original paper. This is down to the fact that the dataset we have is only a subset of the original dataset which only consists of 4 features out of 25 most important ones in the original dataset. On the other hand, Casper surpassed standard NN in test accuracy by 4.5% on average when training with gradient descent and about 7% with genetic algorithm. This is because every hidden units in Casper network is fully trained before adding another hidden unit. Furthermore, each hidden units in Casper network receive connections from all previously added units, whereas neurons in standard NN do not connect to other neurons in the same layer, thus, have weaker feature detection ability.

There are, however, still a lot work need to do. Firstly, neural network may not be the most appropriate solution for small dataset, we could try some traditional machine learning algorithm, like SVM and decision tree, to see whether we can get better test result. Second, batch size is another crucial hyper-parameter we should have tuned. However, since Rprop (must be used with full-batch training) is embedded with Casper, we could not test the result between the two network using "mini batches". Therefore, we may use original Cascor network to compare the standard NN to see if the result can get any better. Lastly, getting more data or meaningful features can almost guarantee a better result in our problem.

5 References

[1] Fahlman, S.E., and Lebiere, C. (1990) The cascade-correlation learning architecture. In Advances in NEural Information Processing II, Touretzky, Ed. San Mateo, CA:Morgan Kauffman, 1990, pp. 342-353.

[2] Kwok, T., and Yeung, D.(1993) A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm. In: Ruspini, H., (Ed.) Proc. Of the ICNN 93, San Francisco, pp. 586-591.

[3] S Baluja and SE Fahlman. "Reducing network depth in the cascade-correlation learning architec ture". In: Pittsburgh (1994).

[4] N.K., T. and T.D., G., (1997). A cascade network algorithm employing progressive rprop. the university of new south wales.

[5] Rahman, J., Gedeon, T., Caldwell, S. and Jones, R., (2020). Brain Melody Informatics: Analysing Effects of Music on Brainwave Patterns. The Australian National University.

[6] Zhang, Y., (2018). Utilizing of Casper Algorithm in CNN for Digit Recognition. Australian National University.

[7] Vitaly, B., 2021. Understanding RMSprop — faster neural network learning. [online] Medium. Available at: <https://towardsdatascience.com/understanding-rmsprop-faster-neural-network-learning-62e116fcf29a> [Accessed 2 September 2018].