Detecting Depression with Various Neural Networks

Eric Hall[u5778954]

Australian National University, Acton ACT 2601, Australia

Abstract. Zhu et al created a neural network that could determine the severity of someone's depression based on the preprocessed physiological signals of a person watching that person talk [1]. I recreated this neural network. I then applied the methods of constructive cascade networks [2] to solve the same problem. Then I applied LSTMs to the raw timeseries data. None of my models achieved great success. The vanilla neural network and cascade neural network achieved slightly better results than the participants in the survey, who themselves achieved slightly better results than would be expected of random chance, but this resulted in only approximately a third of the accuracy of the models in Zhu et al. The LSTM-based model wasn't even able to achieve results better than chance.

Keywords: Depression · Neural Networks · Constructive Cascade Networks · LSTM.

1 Introduction

1.1 Dataset

Zhu et al collected data on people's physiological responses to watching videos of people at various levels of depression [1]. They took measurements of participants' Skin Temperature (ST), Pupil Dilation (PD), and Galvanic Skin Response (GSP). In order to reduce the impact of the fact that different people may have different physiological reactions, they normalized each set of measurements to fall between 0 and 1. They then extracted various features from these measurements, including the min, max, mean, standard deviation, variance, and more. They also knew the self-reported depression level of the people in each video, according to the Beck Depression Inventory - II; these would be regarded as the "true" target values. These depression levels fall into four categories: no depression, mild depression, moderate depression, and severe depression. I was provided with the preprocessed dataset that they obtained in this way. I was also provided with the raw data that they obtained.

1.2 Application of Neural Networks

By applying a feed-forward network to the preprocessed data, Zhu et al were able to predict the level of a person's depression with 88% accuracy, judging by the physiological signals of the person watching a video of the potentially depressed person [1]. By also applying a genetic algorithm to select the features to be used by the neural network, they were able to improve this to 92% accuracy.

1.3 Extensions to the Dataset

Some work has been done that builds upon the work of Zhu et al. One study created a similar dataset, which contains all the types of measurement used in the original dataset, but also the Blood Volume Pulse (BVP), which "indicates the volume of blood running through the vessels over time" [3]. Similar methods have also been used by researchers to detect deception, as opposed to depression [4].

1.4 Constructive Cascade Models

Khoo & Gedeon proposed two new constructive cascade algorithms called Local Feature Constructive Cascade (LoCC) and Symmetry Local Feature Constructive Cascade (SymLoCC)[2]. These are kinds of convolutional neural networks, used for classifying faces. In these methods, an initial convolutional cascade layer is trained that, when given a picture of a face, predicts whose face it is. Then additional cascade layers are repeatedly added and trained, each of which takes as input both the input to the model as a whole as well as the output of previous cascade layers. The outputs of each additional cascade layer are added to the output layer.

1.5 Long Short Term Memory (LSTM) Neural Networks

In recurrent neural networks, some internal state is maintained between time steps. We can think of this internal state as being produced by the network at any given time step, and passed on to another layer of the same network at the next time step. Then over time, these layers stack on top of each other to create a very deep neural network. Then in a sense, recurrent neural networks can be considered a kind of deep learning [8]. This means that they share some similar characteristics, for example, the vanishing gradient problem [9][10].

Long short-term memory neural networks are a kind of recurrent neural network that solve the vanishing gradient problem in recurrent neural networks [9].

2 Method

2.1 Problem to be Solved

We use Zhu et al's pre-processed dataset, and just like in Zhu et al's research, we aim to use the physiological signals of the observer to detect the level of depression of the person in the video they are watching. This is a worthwhile problem to solve, because depression is a major problem that a large number of people in our society face. The ability to detect depression would be a significant help in allowing people to recognise depression in themselves and/or others, and it would allow them to seek professional help in overcoming it. If models like the ones discussed in this paper were to be applied in the real world to detect depression, it would have the potential to improve millions of lives. This is why the aforementioned problem is an interesting and worthwhile one.

2.2 Purpose of my Research

The purpose of my research is to compare the effectiveness of three different models in solving the above problem. The first model aims to replicate the vanilla neural network used in Zhu et al [1], the second model is an application of the techniques of Khoo & Gedeon [2] to the preprocessed dataset of Zhu et al, and the third model is an application of LSTMs to the raw data obtained by Zhu et al.

2.3 Why the Second Model is Interesting

The two research papers do not deal with the same type of data, nor do they solve the same problem. Whereas Khoo & Gedeon's research deals with images and aims to recognise faces, Zhu et al's research deals with various statistics describing physiological signals and aims to detect depression. In spite of the differences between the two papers, there are also some similarities. Both papers aim to classify data into some category (in the case of Khoo & Gedeon's research, a category corresponds to a person, to whom the face belongs, whereas in the case of Zhu et al's research, a category corresponds to a level of depression), and both papers aim to train neural-network-based models to do this.

Due to both the similarities and the differences, it is interesting to consider what would happen when techniques from one dataset are applied to the other dataset. In particular, what advantages and disadvantages do constructive cascade models have, when compared to feed-forward neural networks, when applied to non-image-based data? Will they be more effective than the methods used in Zhu et al's research?

2.4 Why the Third Model is Interesting

In the process of preprocessing their data, Zhu et al reduced timeseries data to a series of statistics about the data, including the minimum normalised GSR, the maximum normalised GSR, and many other features [1]. Reducing such a large quantity of data to such a relatively small quantity of data has the inevitable consequence that a large amount of data will be lost, and it is almost certain that some of the data that is lost will have been potentially useful data. Therefore, it is interesting to consider models that take into account all of the collected data, to see if more useful information can be extracted from the data, so as to aid in classification.

Since the input data is timeseries data with no fixed length, it makes the most sense to apply recurrent neural networks of some sort to the data. Specifically, LSTMs are a very commonly used kind of neural network that have proven themselves to be effective.

2.5 Feed-forward Neural Network

My feed-forward neural network aims to emulate the neural network used in Zhu et al as much as possible. For this reason, I chose to have a single sigmoid hidden layer with 50 neurons, using all the features of the preprocessed dataset, with 4 outputs representing the 4 depression levels, trained with the Adam optimizer and Cross-Entropy loss.

In addition to just being the settings used by the original paper, these settings make sense for this problem (as you would naturally expect from the settings chosen in a research paper). The Cross Entropy Loss is designed to be used to predict categorical data, which is precisely what we are doing. When classifying categorical data, it is natural to have as many outputs as categories, so that each output corresponds to a category. In this specific scenario, a reasonable alternative would be to just have one output that ranges between 0 and 3, where each integer denotes a depression level, because the depression levels follow a nice, linear, relationship, where "no depression" is less severe than "mild depression" which is less severe than "moderate depression" which is less severe than "severe depression". But having one output for each category is a completely reasonable representation.

I also kept the learning rate at its default value of 0.001. I did this because if you look at figures 1 and 2, it seems that the neural network is learning at a reasonable rate using this default value. I chose to train the neural network for 100 epochs. I did this because I found that training the network for much more than that would lead to visible overfitting in the training loss vs testing loss graph for all validation folds.

A setting that might be improvable is the set of features that are input to the neural network. Many of the features contain similar data to each other, and therefore may be redundant [1]. It might be a good idea to select only a subset of these features to use in the model, as was done by Zhu et al. This would reduce the number of weights in the model, reducing the amount of work necessary to train it. However, determining the best features to remove is a non-trivial problem and it is much simpler to just include all the features, which is why that is what I did.

2.6 Applying Constructive Cascade Models to the Dataset

In the second part of my research, I applied the techniques of Khoo & Gedeon [2] to the dataset of Zhu et al [1]. The problem of classifying depression levels based on various statistics is substantially different from the problem of face recognition using images, and so substantial changes needed to be made to the method in order to make it work. Most notably, convolutional neural networks were used in the face recognition problem, which works very well when applied to image data, but doesn't work very well when applied to non-image data. I had to replace the convolutional neural networks with sigmoid neural networks.

Otherwise, the Constructive Cascade model I used was quite similar to the model used in the paper. A linear model is trained on the input and output. After it has finished training, additional layers are repeatedly added and then trained, so that only one layer is being trained at a time. The input to each additional layer consists of both the input to the model as a whole and all the outputs of all the previous layers. The outputs of all the layers are added together to provide the output of the model as a whole. During training, I again used the Adam optimizer and Cross-Entropy loss, so that my results would be comparable to those obtained with the feed-forward network.

2.7 Labelling the Raw Data using the Preprocessed Data

In the raw data I was given, the videos were simply labelled "video_1", "video_2", etc, and they did not have any indication as to the depression level of the subjects of the videos. In order to work out these depression levels, I normalised the GSR using the same method as in Zhu et al [1] (I applied a min-max scalar on a participant-by-participant basis), I took the minimum over the normalised GSR timeseries, and compared the results to the preprocessed data, to see which videos matched with which depression labels.

I noticed that for many participants, the normalised minimum GSRs I calculated matched up precisely with the normalised minimum GSRs in the preprocessed dataset, in exactly the same order. For several other participants, the normalised minimum GSRs I calculated did not match up precisely with the normalised minimum GSRs in the preprocessed dataset, but they still correlated very closely with each other. This indicates that the videos in the raw dataset were in the same order as the videos in the preprocessed dataset. I was able to use this information to label the raw data.

Since for some, and only some participants, the normalised minimum GSRs I calculated were different to the normalised minimum GSRs in the preprocessed dataset, we know that there must have been an error at some point in either Zhu et al's or my work.

2.8 Making the Sampling Rate Consistent

In the raw data, the pupil dilation data was recorded with a sampling rate of 60Hz, while the other data was recorded with a sampling rate of 4Hz [1]. In order to make the sampling rate consistent, the average of every consecutive set of 15 observations from the pupil dilation data was taken, which artificially lowered the sampling rate to 4Hz, consistent with the other data.

2.9 Applying LSTM Neural Networks to the Raw Data

Now that the raw data had been labelled and the sampling rate had been made consistent, I could use it to train and test an LSTM neural network. I used pytorch to do this. Like in the vanilla neural network, the output size was 4, so as represent the 4 depression levels. But unlike the vanilla neural network, the input size was 4, so as to represent the GSR, left pupil dilation, right pupil dilation, and skin temperature at any given time step.

To use the model, all the rows of an appropriate timeseries would be sequentially fed into the model, at which point an output could be obtained from the model. If one was training the model, one could then backpropogate the error through the model using pytorch's autograd functionality, and apply some optimizer to the model.

I continued to use the Adam optimizer and Cross-Entropy loss, to keep the results from my three models comparable. I did not change the learning rate from its default value of 0.001, as I could see from figures 5 and 6 that the learning rate was neither too low (in which case the loss would converge too slowly) nor too high (in which case the loss would change in an unstable manner).

2.10 Evaluating my Models

Just like in Zhu et al [1], I used leave-one-participant-out cross-validation to train and test each neural network. Since the physiological reactions of a certain participant to different videos are likely to be relatively similar to each other, it makes sense to group all the reactions of one participant together, and either include them all in the training dataset, or include them all in the testing dataset. So, for every participant, we create a training set containing data from the rest of the participants, and create a testing set containing data from the chosen participant, and we train/test the model on these sets [1]. Each such training/testing session is called a validation fold. For each training and testing set generated in this way, we make sure to re-create/reset the model, so that it isn't tainted from the training it has done on the previous training sets. This allows every piece of data to be used at some point as both training data and as testing data, without ever testing the model on data that the model has been trained on.

Zhu et al primarily used the F1 score to evaluate their models (although they also provided the accuracy, precision, and recall of their models). However, I chose instead to use the accuracy. This is because the main reasoning people use to justify using the F1 score is the fact that accuracy can behave poorly when dealing with extremely imbalanced classes. However, in this dataset, this is not a problem, because there are equal numbers of datapoints in each class.

In order to detect whether or not my models were overfitting, for each validation fold, I recorded the progression in training loss and testing loss over time. I used this information to create graphs comparing training loss to testing loss.

2.11 Attempts to use Jitter to Reduce Overfitting

With regards to the Constructive Cascade model, in an attempt to reduce overfitting, I added noisy copies of the dataset to the dataset, in the hopes that the algorithm would need to generalize in order to deal with the noise. This is a technique used by some researchers to improve the generalization of neural networks [5], however some other researchers cast doubt upon the effectiveness of this technique [6] [7]. It unfortunately didn't bring me much success.

3 Results and Discussion

Table 1. Accuracy of various methods. My data is averaged over all the validation folds.

Neural Network	Constructive Cascade	LSTM Model	Participant Guesses [1]	NN by Zhu et al [1]	GA + NN by Zhu et al [1]
0.32	0.30	0.23	0.27	0.88	0.92

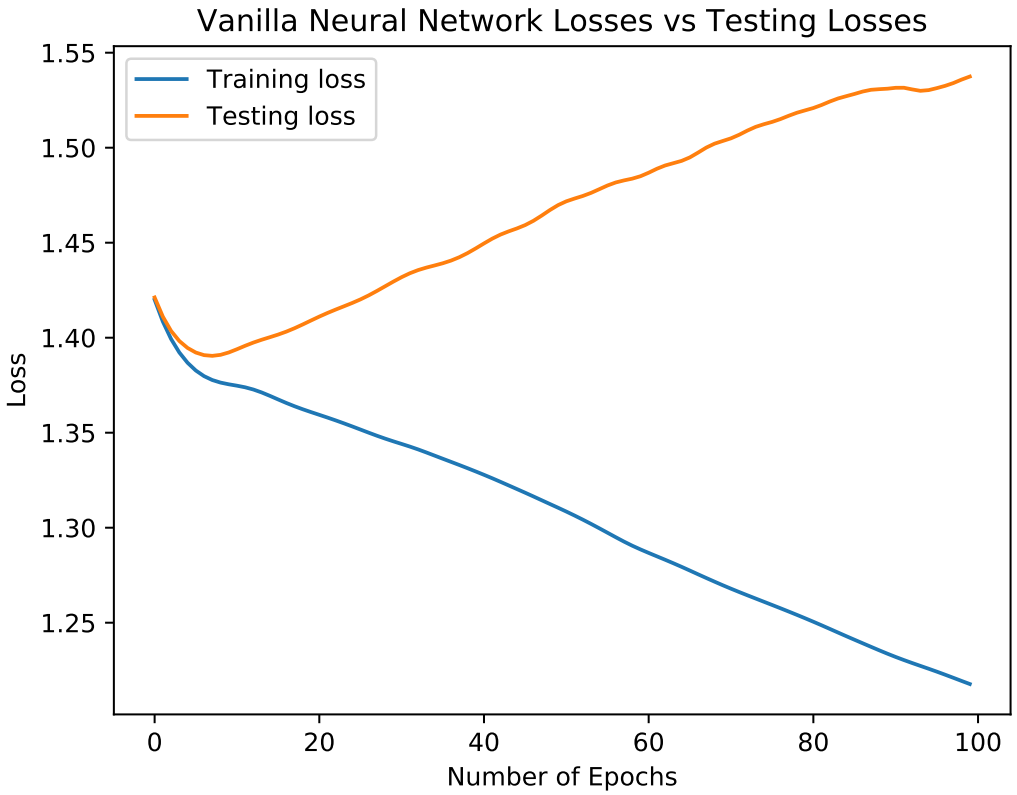

Fig. 1. Training loss vs testing loss for one of the validation folds when training the vanilla neural network. Notice that high levels of overfitting start occuring from the very start of the graph.

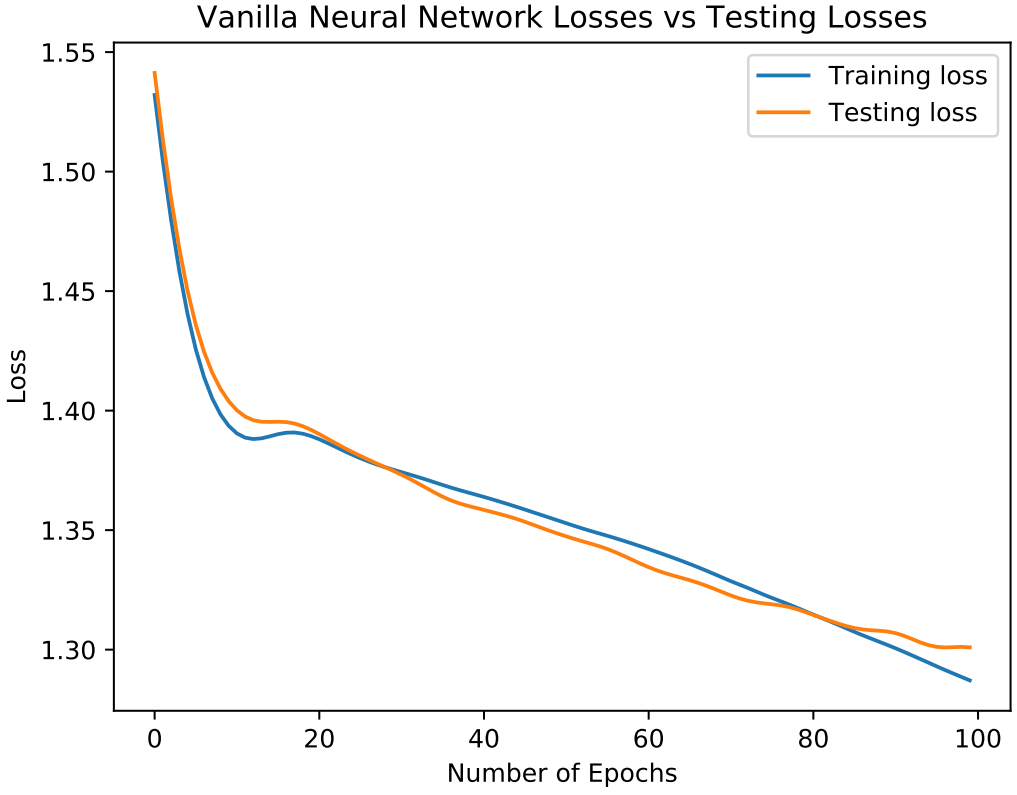

Fig. 2. Training loss vs testing loss for a different validation fold, when training the vanilla neural network. Notice that overfitting doesn't start to occur until possibly the very end of the graph.

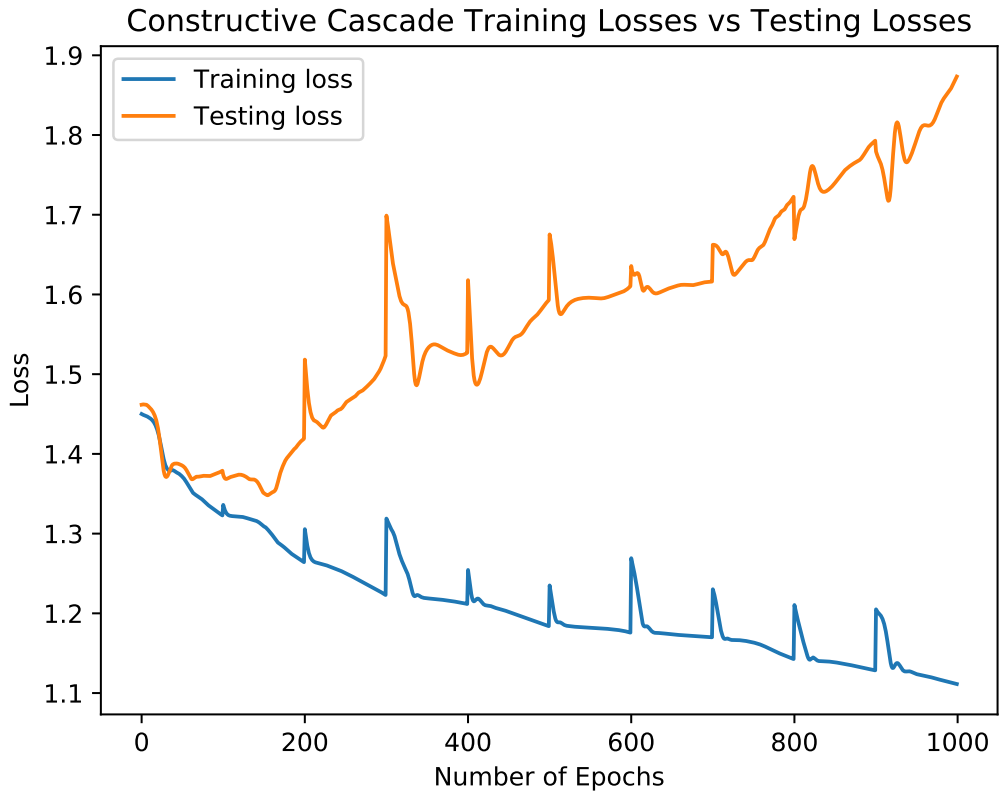

Fig. 3. Training loss vs testing loss for one of the validation folds when training the constructive cascade model. Notice that high levels of overfitting start occuring from the very start of the graph.

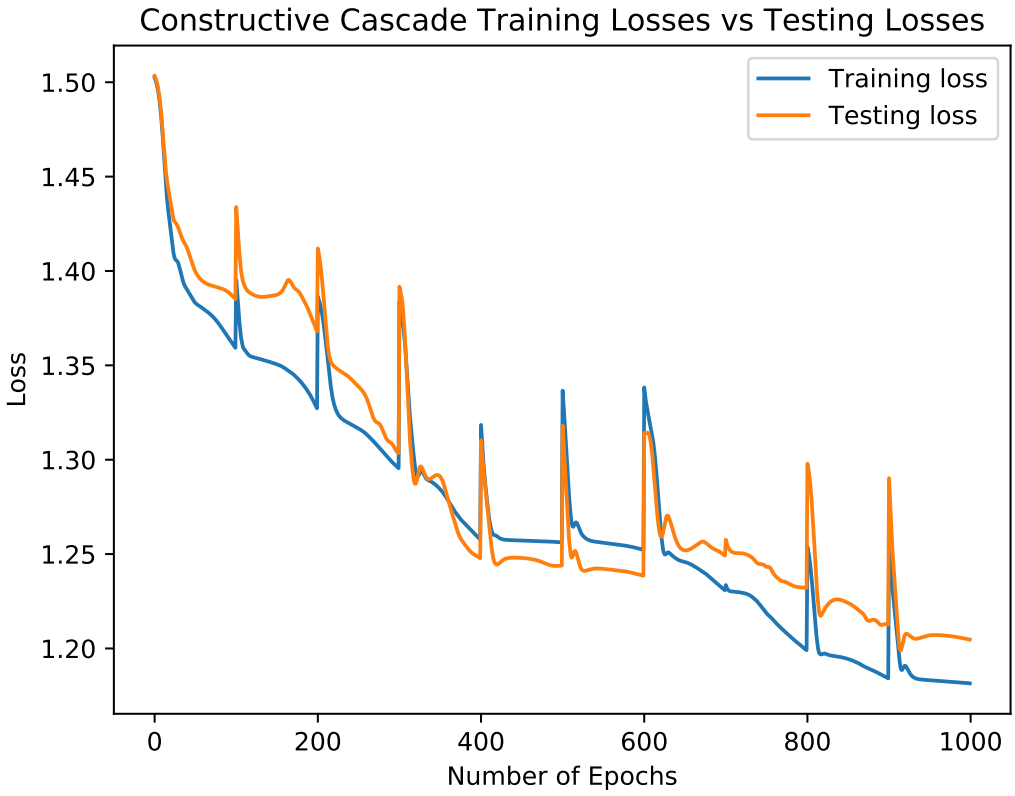

Fig. 4. Training loss vs testing loss for a different validation fold, when training the constructive cascade model. Notice that overfitting doesn't start to occur for the duration of the graph.

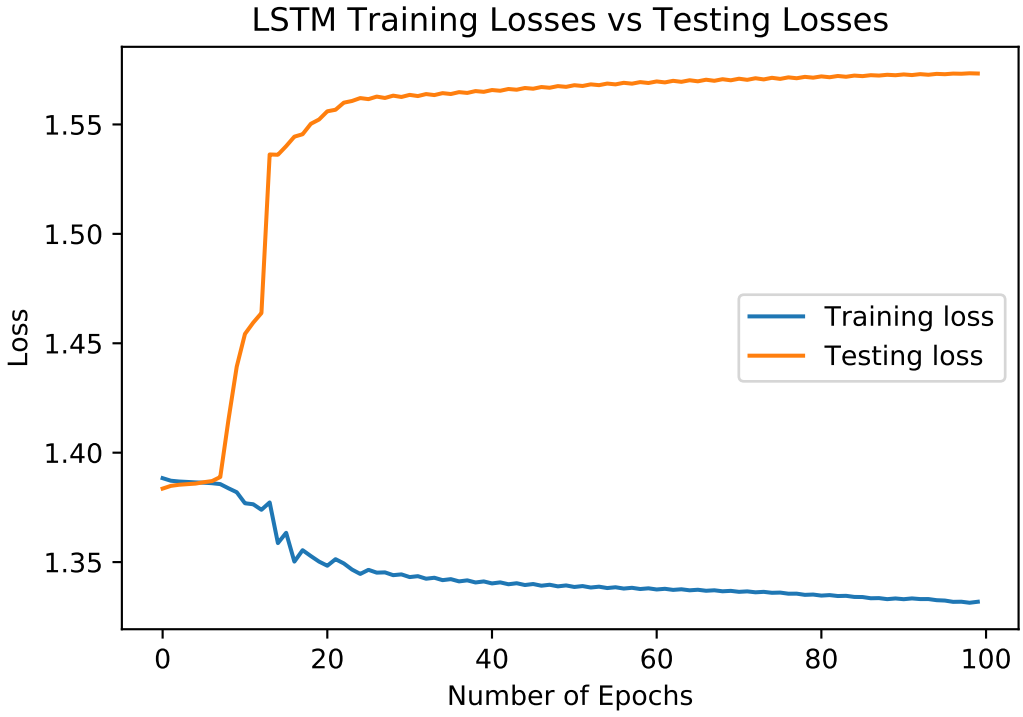

Fig. 5. Training loss vs testing loss for one of the validation folds when training the LSTM model. Notice that high levels of overfitting start occuring from the very start of the graph.

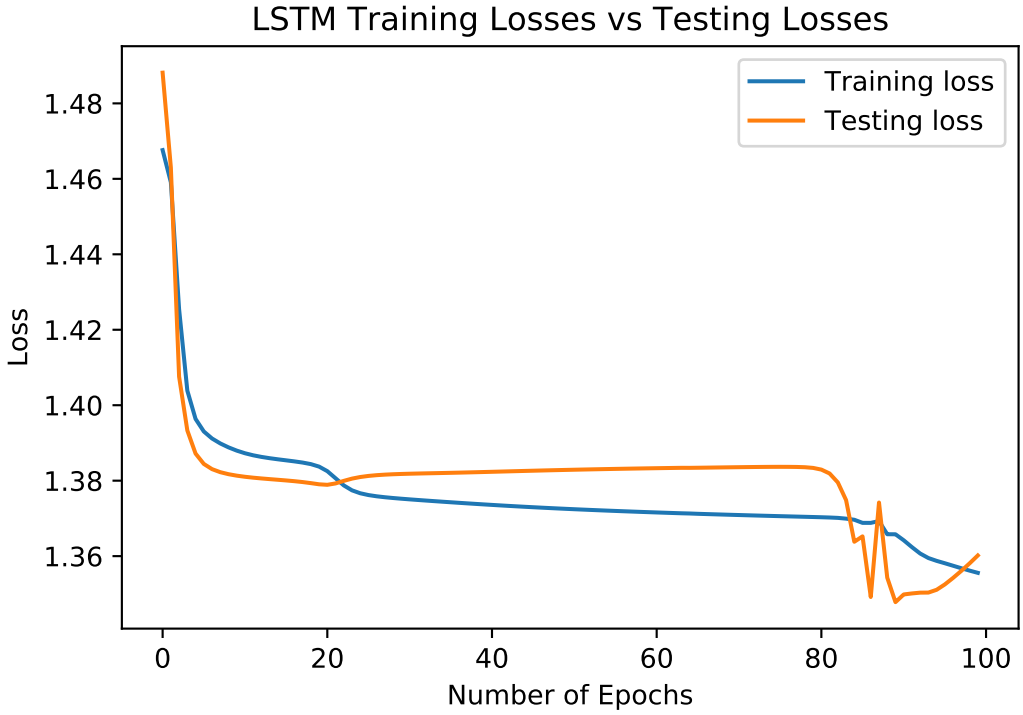

Fig. 6. Training loss vs testing loss for a different validation fold, when training the LSTM model. Notice that overfitting doesn't start to occur until possibly the very end of the graph.

What is most noticeable about these results is that I was unable to come close to achieving the level of accuracy claimed by Zhu et al [1]. However, I was able to get results that were substantially above chance, and slightly above the accuracy of the predictions of the participants. My constructive cascade model failed to be an improvement upon my neural network model. In fact, it is slightly less effective, although still slightly above the accuracy of the predictions of the participants. My LSTM model failed to even obtain an accuracy above that of random chance.

By training an LSTM on the raw data, I was able to find out for myself why Zhu et al had extracted various statistics from the data rather than using neural network techniques directly on the raw data. The main problem was that it took an extremely long time to train the LSTM model, on the order of several hours. Extracting statistics from the data allowed Zhu et al to both reduce the total amount of data and also reduce the amount of time required to forward-propogate and back-propogate through the neural net, which reduced training time. Not only that, but extracting features in this way also seems to improve classification accuracy, perhaps because it makes useful features visible that it would have been hard for the neural network to extract on it's own.

My vanilla neural network model was intended to be a replica of Zhu et al's model. The fact that my results do not align with Zhu et al's results indicates one of three things: Either there was some mistake in my experimental method, there was some mistake in their experimental method, or for some reason, my model was not a sufficiently faithful recreation of their model, despite making no actual mistakes in its implementation. I only had a short deadline by which I had to conduct this research, whereas Zhu et al appear to have written multiple papers on related topics and put a lot of work into their research. Therefore, it is most likely that the mistake is on my part, or that I failed to faithfully recreate Zhu et al's model. However, the possibility that there is some mistake in the research of Zhu et al is not to be completely discounted.

In the analysis of figures 1 through 6, we need to keep in mind the following. If the training loss decreases while the testing loss increases, that indicates that overfitting is occuring, as the model starts to predict very well on the training data but fails to generalise to new data. If both the training loss and testing loss decrease at the same time, that indicates overfitting is not occuring, since the improved classification probability on the training data does generalise to new data. Then figures 1 through 6 show a general pattern in the levels of overfitting in our models. Looking at these figures, we see it is universal amongst my models that in some validation folds, overfitting will start occuring as soon as the model begins training, whereas in other validation folds, overfitting will not start to occur for a substantial amount of training time. This makes it difficult to make sure that the models are trained for an appropriate amount of time, as the ideal length of time varies from validation fold to validation fold.

There are also qualitative differences in the ways in which the losses of different models change over time. In the constructive cascade network, there was a loss spike every time a new cascade layer was added to the data, since it took time for the new layer to learn the data. In contrast, the changes in loss for both the vanilla neural network and the LSTM model are quite smooth.

The LSTM seemed to finish training in a far smaller number of epochs compared to the other two models. After the first 10 or so epochs, there was little difference in the training/testing loss over time. However, this is tainted by the fact that the LSTM took far longer to complete each epoch took than the other models, as well as the simple fact that the classification accuracy of the LSTM was poor.

4 Conclusion

The vanilla neural network and constructive cascade models I trained were slightly more accurate than the guesses of the participants, which were themselves slightly more accurate than chance. However, they were far, far less accurate than the models in Zhu et al [1], with approximately a third of the accuracy. The constructive cascade model had a slightly lower accuracy than the neural network model. The LSTM model failed to achieve an accuracy above that of random chance.

4.1 Future Work

My models appear to be overfitting significantly to the data, and this is adversely impacting the results. Further work in the area of reducing the overfitting of the models would be valuable. Specifically, it might be a good idea to gather additional data to train the model on, somehow simplify the model in order to reduce the number of parameters, or experiment with various regularisation techniques.

An interesting extension to the research of Zhu et al [1] would be to work out how little information must be given to an observer so that they give off recognizable physiological signals that indicate the level of depression of the person being observed. In particular, if a person is only given a video of someone walking down the street, is it possible for them to subconsciously recognise that person's level of depression, and for that information to be picked up by a neural network reading their physiological signals?

References

1. Zhu, X., Gedeon, T., Caldwell, S., & Jones, R. (2019). Detecting emotional reactions to videos of depression. In IEEE International Conference on Intelligent Engineering Systems.
2. Khoo S., Gedeon T. (2008). Generalisation Performance vs. Architecture Variations in Constructive Cascade Networks. In International Conference on Neural Information Processing.
3. Zhu, X., Gedeon, T., Caldwell, S., Jones, R. (2019). Visceral versus Verbal: Can We See Depression? In Acta Polytechnica Hungarica.
4. Gu, X. (2019). Detecting the Doubt Effect and Subjective Belief Using Neural Network and Physiological Signals. Published on the ANU's Computer Science Individual Projects Website.
5. Zur, R., Jiang, Y., Metz, C. (2004). Comparison of two methods of adding jitter to artificial neural network training. In International Congress Series.
6. Gorp, J., Schoukens, J., Pintelon, R. (1998). Adding Input Noise to Increase the Generalization of Neural Networks is a Bad Idea. In Intelligent Engineering Systems Through Artificial Neural Networks.
7. Gorp, J. (2000). Using Variability to Analyse the Effects of Adding Jitter. In International Joint Conference on Neural Networks.
8. Pascanu, R., Gulcehre, C., Cho, K., Bengio, Y. (2014). How to construct deep recurrent neural networks. arXiv preprint.
9. Hocheriter, S., Schmidhuber, J. (1997). Long Short-Term Memory. In Neural Computation.
10. Nielsen, M. (2015). Neural Networks and Deep Learning. Determination press.