# Stimulus Prediction Using Bidirectional Neural Networks: Using Bidirectional Neural Network to Develop Prototypes of Anger

Amelia Poetter[1]
U6054910@anu.edu.au

[1] Research School of Computer Science, Australian National University

**Abstract.** Using the observer responses to an emotion has been shown to be highly effective in determining its veracity[1][2]. This project used the dataset from [2] to investigate if this can be expanded upon by using different classes of prediction on the same dataset. Furthermore, we explored using simple neural network, a bidirectional neural network and an LSTM on the timeseries version of the data, in attempting to predict the video being viewed. For the BDNN, we also try to determine representative class patterns. The performance of the basic neural net was significantly above chance, however not precise enough to be used for prediction. The BDNN performed terribly, with only slightly above chance results and similar for the LSTM.

**Keywords:** Bidirectional Neural Network, Artificial Neural Network, Pupillary Response, Physiological Signals

## 1 Introduction

There are many applications where determining from physiological signals the class of emotion that someone is observing is desired. The scope of this project is exclusively examining anger. We investigate methods used to analyse the how pupillary response is affected by a video of someone demonstrating or expressing anger, with the implication that this could be sub-categorised into more useful classes and used in a range of scenarios. For instance, instead of analysing the videos under the lens of its veracity; with more domain knowledge and information on the specificities of the videos used in the experiment, one could separate the videos into more useful and larger classes determined by their content as implemented by the experimenter. It would likely be highly beneficial to further subsample the dataset and find interesting representative features of more specific subclasses of anger, such as passive or assertive (or many more.) One can imagine this being useful in the development educational virtual realities and avatars, to be able to effectively or even instantly predict the style of anger that is being portrayed on the screen to optimise this to set specifications. Another possible approach, would be as a learning tool for actors, if something could be set up that allows them to easily assess how their portrayals are being perceived by the viewers as the actor practices or performs.

In this study, we explore neural network approaches to classification of anger subtype based on an observers pupillary response. This work builds heavily on prior work we conducted that specifically examined the BDNN framework proposed by [3]. In that work, we attempted to find representative vectors of features for the different videos in the dataset. The thought being that pupil dilation is a very strong predictor of the class of video being viewed at least. The hypothesis was that perhaps given the incredibly strong relationship, more detailed relationships could be extracted regarding not only the veracity of the anger the participants views, but other features of anger evident in the videos. For the scope of this project, such 'features' were substituted with just each individual video, given lack of knowledge about what the videos were specifically displaying.

The work is expanded on in this paper, by exploring LSTM implementations on an extended timeseries version of the previous dataset. LSTMs are of particular interest as the data is of a time series structure naturally. It is therefore highly likely that there is rich information contained within the series that could be used to further analyse this style of problem. Moreover, LSTM offers bidirectional functionality which is interesting to this project for the same reasons that a BDNN is considered and has been discussed. The similarity of the features that these two models offer adds an additional layer to this task, that it can allow the comparison of these datasets.

Ultimately, the simple neural network produced more effective results than the other two designs. A few reasons for this that we speculate may be the cause, include: ..

## 2 Method

We investigate the relationship between the features measured of pupillary response to predict specific videos from of the dataset.

The process of validating the hyperparameters for the basic neural net model in this project was investigated as an important decision in the overall building of the mode. An important consideration of this step is that the goal of this net is not to optimise it, but instead to appropriately build a model that can be compared to the BDNN as we wish to compare performance of the two. Additionally, the BDNN requires significantly more epochs to converge and it was found to be infeasible to tune this model (particularly for the cluster centroid finder that is discussed in more detail in implementation of BDNN) and select parameters (such as max depth) using this. Given these two important consideration, both the basic neural network model and the BDNN were trained using Supervising Gradient Descent. This involved plotting the models and manually adjusting the maximum number of epochs by inspection of change in loss of training and test set with a test ratio of 20%, equaling 80 records in total. Multiple seeds were used in the development of the model, and variation in the models performance was analysed. It was found that there was very little fluctuation in the results, so the method was retained.

An inspection of the feature variables was done and it was determined that in their raw for they had already been pre-processed in a way that was highly effective on the dataset so no further alterations were made to them in terms of scaling and normalisation. Further analysis of the inputs was necessary to ensure that the features were inputting relevant information to the model and reduce data redundancy for the benefit of the models performance. Finally, the output vector 'Video' is encoded with one-hot encoding, given its categorical nature and this forms the target for the neural nets analysed. These specifications apply to both the basic neural network and the BDNN implementation.

The model was built around the 6 input features received from the anger dataset. These are labelled 'Mean', 'Std', 'Diff1', 'Diff2', 'PCAd1' and 'PCAd2', they represent the pupillary signals of the observers as they view the anger demonstration. The input features are all numeric and it was found that no further normalisation with min-max scaling improved the models. Additionally, the output feature for this model is the 'Video' label which is encoded with one-hot encoding to be useful for classification. The LSTM portion uses a different format of data that is explained in more detail in that section.

### 2.1 Implementation of a Neural Network

One hidden layer was used, as this is the same as the technique for a BDNN and so for comparability the same was used here. Manual inspection of the model performance determined the optimal hyper-parameters to be a ReLU hidden layer activation function with a maximum number of epochs was 150, a learning rate of 0.01, and 50 hidden neurons.

### 2.2 Implementation of a Bidirectional Neural Network

The Bidirectional Neural Network Model [3] was applied to the dataset to investigate if it can effectively predict the video class of the dataset being investigated additionally, it offers the functionality that it can be modelled in the reverse direction, so features can be determined from class as well. This model is then analysed against the neural net as discussed above and the differences and performances were analysed and compared to determine if there was benefit in applying this technique. Furthermore, we utilise the nature of the BDNN in determining the investigate whether we can extract meaningful patterns, or key features.

A BDNN works by taking a one to one feature association of the data and using a back propagation algorithm on the model. For the purpose of this model, the input features are the 6 features described aggregated for each video. The method for aggregating is discussed in more detail below and in the discussion. Each of these input patterns is then mapped onto only one output node (the relevant video classification.) In following with the

technique described, there is one hidden layer, whose number nodes were optimised with manual inspection and finalised as 50. The model is trained normally in this direction for the first 50 epochs, using back propagation to update the weights for the hidden layer and the output layer.

After 50 epochs, the training direction is reversed. This means that the weights are retained from the forward direction, however the output weights now serve as the hidden weights and conversely, the output weights are determined by the hidden weights. The hidden bias for this model is also retained, and the output bias of the forward direction is ignored and a separate output bias for the output layer is generated. At regular intervals, as determined by the reduction in loss or when the set epochs have been reached the method of training is reversed following the procedure described above with the obvious modifications made depending on which direction the model is being converted from. Additionally, in the first pass in the reverse direction, the model generates the output bias that is not used from the forward pass. However, for both the forward and backward in subsequent passes through the model, the biases here are retained from the previous training on this direction.

Of further significance in this particular implementation of the BDNN model is that the loss function is computed differently and depends on the direction of the training. For instance, in the forward direction, the network is outputting the video prediction in the form of a one-hot encoded class vector. For this instance, cross entropy classification from the PyTorch package is used as it is effective for classification and was found to perform well in this use case. Contrariwise, the reverse direction seeks to generate the likely features of an input class (note in this instance input refers to the output of the previous direction, the video identifier.) Cross entropy Classification would not work effectively in computing loss here, so an alternative had to be used. Experimentation was done and ultimately it was clear that the mean squared errors, which sums the square of the difference between the points predicted by the BDNN and the true points for the features.

The optimiser used to for the gradient descent was Adam, again, this was implemented using the PyTorch library. In following with the technique paper and the basic neural net described above, the model was analysed using a constant learning rate, that remained the same in both directions. As the technique paper mentioned, it is possible that the performance of the model could be optimised by using dynamic hyperparameters such as these. The stochastic gradient descent algorithm performed poorly on the model and so was not used. As discussed above, a manual analysis of how the model performed on both training and test sets was used in the decisions of the implementations of these features. More details on these decisions can be found in the discussion.

For first step for training a BDNN as content addressable memory is to transform the training data so that it resembles a one-to-one function. This allows the model to adjust and be trained in the reverse direction. The dataset of course, had many input patterns for each class (for each video, data is collected from multiple participants and the model must be trained on some kind of aggregate of these pieces of information.) The statistical manipulations of the input patterns were experimented with, specifically taking the mean and median of the input features for each pattern. Additionally, a BDNN Model was trained to find the centre of the class clusters. The inversion of the training data was selected by training and fitting the model and comparing metrics (accuracy and time performance). To build the cluster centroid finder with the BDNN the model is trained with no restriction on the relationship, according to the method outlined and then a reverse pass is run to predict the representative vector for each video type. For a further level of analysis on how effective the representative patterns were, the cosine distance was used to determine which of the videos representative vector was the closest to each feature pattern of the dataset and then the proportion of the correct classification of these was looked at. The constant learning rate is in keeping with the technique from the paper, it was manually tuned to be 0.01.

### 2.2 Implementation of a Long-Short Term Neural Network

The Long-Short Neural Network Model [5] was applied to the extended time-series dataset to investigate if LSTM is a viable technique for pupillary signal investigations and classification. To determine its effectiveness, an LSTM model was built using an extended version of the dataset used in the first two Neural Networks. This process can be summarised as pre-processing and exploration of the new dataset, training of the LSTM and testing and comparison. For further analysis, the times-series LSTM is used as a binary classification task to predict if observers are viewing a video of real or fake anger. The model was shuffled and then separated into training and test sets at random.

The new dataset takes into account the time-series nature of the data that has been collected, meaning that instead of training a learning model specific features that have been extracted from a time-series data collection (e.g. ‘mean’), the training data is the dilation of the pupils at each time step of the viewer watching the video. The raw

format of this data comes is separated by pupil (left or right) and is then further split into a table per video that has a column for each participant that watched the video, which contains the time-series data for watching that video as described above. Some precursory steps were taken to ready the data for training with an LSTM. Firstly, for the purpose of our research, we were not interested in each pupil separately, although that is perhaps an area of future interest. To combat this, each participant/video combination was matched up from the left and right pupils and subsequently, the mean was taken at each timestep. Secondly, the time series of each video were of unequal length, making them unsuitable for an LSTM model. To combat this, we used padding with zeroes at the end of the longest series. This seems to be the convention for most LSTM applications, however in future we would like to explore the both truncation as an alternative and further to experiment with both of these techniques at the beginning of the sequence. For this project, we assumed that the beginning of the sequence is of more interest.

The training stage involved the initial building of the model, and the optimisation of parameters and hyper-parameters. In keeping with our interest in comparing the efficacies on our pupil data, steps were taken to ensure a reasonable degree of similarity to the modelling approach. Thus, A vanilla many-to-one LSTM was implemented using the PyTorch package within Python. the layers of the LSTM were kept minimal using the default PyTorch settings, to allow for more specific comparisons to be made to the two other models. Of note about this technique is the batch style design which used a moving window of a fixed size and iteratively trains on different portions of the training set. The parameter determining how large this window is, is one of the hyperparameters we tuned in this task, and we refer to is as batch size. Experimenting with the models ability to achieve accurate ratings, we adjusted various hyper-parameters and settings and compared their results. The parameters we adjust are learning rate, the loss function, the output activation function, the number of hidden layers and the optimiser. To optimise the model, settings and hyperparameters were measured against the testing data. More detail of this process can be found in Results. In order to inspect the effectiveness of the LSTM implementation, the model was explored with a two class version of the classification using the video labels of ‘True’ or ‘False’ and the results were compared.

The model was then analysed against the two neural net models discussed above. The differences in performance and accuracy were analysed and compared to determine if either was a viable technique for this class of problem.

## 3 Results and Discussion

A limitation of the BDNN model for this project is also that the nature of the features are not that meaningful in themselves, so given the difficulty that there seems to be in classifying, it is possible that this is because some of the relevant vectors are actually highly similar, and if we had properly classified the video types the model would perform much better. Additionally, each feature was removed from the dataset as a visual inspection of the variables with scatter plots determined that there were some strong correlations between the datasets and so the model was tested with the removal of each feature iteratively. In each adaptation the results were extremely poor.

For the loss function in the reverse direction of the BDNN, it was determined that the MSELoss package from PyTorch would be the most suitable. This is definitely one of the weaknesses of the BDNN, as it doesn’t effectively factor in that the true intention of the model in this direction is not so much to make sure that the model generates values for each of the inputs that is close to the target values for that class, but to also ensure the separability of the feature representations of each class in respect to the other classes. Additionally, perhaps a loss function with a more vector-like representation would be beneficial so as to create more distance between the outputs and allow for more accuracy in using the representative patterns. More improvements could be made by updating the weight of each feature in terms of its significance in producing the loss. Some preliminary analysis demonstrated that there is some overlap in the features and it is of course likely that there could be significant optimisation of the algorithm if this was remedied, or experimented with. The MSE also addresses the nature that there is

The representation of the classes in the BDNN posed a highly significant challenge in developing a model that was able to perform accurately on the dataset. Different measures of generating this standard feature pattern for the classes were explored. The types of these included a statistical analysis, done by calculating the median, mean

and z-score of the input features and also using a separate BDNN to generate expected feature values for the each video. The analysis of these features was done in two ways. Firstly, by training and testing the BDNN with the values of the representative class and exploring the results. And additionally by, using a cosine similarity metric to evaluate for each pattern in the dataset, which of the representative class patterns generated from each of the methods described above, yielded the most correct classifications as determined by the target value 'Video' from the dataset. Firstly the dataset was used in its full to generate these for that

As expected from the disappointing results found in the initial BDNN's that were trained on this dataset, the BDNN did not perform well when modelled with the representative clusters generated from the BDNN. The research conducting for this project did not manage to achieve a reliable way of generating the representative class patterns for the model. This would need to be solved in some way for the techniques implemented on the dataset to have any reliability or use in prediction or proto-type generation as discussed in the above sections. Without information about the specific videos, and without having watched them and consequently, not having the domain knowledge for a complete analysis as to what the effect of the videos would be on the separation into representative patterns, it is possible to exhaustively explore the desired range of options. It is likely that there is not enough significant difference between the videos form the dataset to produce a meaningful separation, however there is some possibility that they may be split into larger subsets of the classification of the

An alternative approach would be to reclassify the videos into larger subclasses. The effectiveness of this approach obviously depends on the videos if this is a viable or effective strategy to attempt to implement. However, without additional data, this is the most likely option to produce useful separation of the classes and allow the BDNN to train effectively. Furthermore, this report was primarily an investigation of the viability of a technique that would ultimately be useful on this dataset to perform more varied and specific classification. Consequently, it is highly unlikely that videos selected on the basis of if they represent an angry emotional depiction that is either genuine or fake would naturally be individually relevant to any other features that you are investigating the response to. The example responses as discussed in the introduction section of this response, for instance separating passive aggression, open aggression and assertive anger. It is hard to imagine a further classification of anger into the 20 subclasses that this dataset conducted. This pipeline approach of a manual sorting of the videos into a smaller number of classes that have a naturally higher number of samples would reduce noise, probably be more useful, and also likely improve the accuracy and it would be interesting to see if the BDNN specifically would manage to perform with higher accuracy on different subsamples. Another approach to this idea, would be that there is likely to be

The application of an LSTM was attempted to complement and compare with the BDNN. The LSTM offers interesting new features, given that it can handle time series data, as well as be trained in two directions. However, similarly to the BDNN the results were poor and these results indicate that LSTM is not suitable for the data. The performance metrics achieved from all variations of tuning on our LSTM were approximately the same accuracy as random guessing, obviously a poor result. The specific detail of the hyper-parameter and parameter implementations, fig.. contains an overview of these results. There seems to be no parameter of combination of parameters that have any significant influence on our results. For this reason, it seemed highly likely that there is an error in the LSTM we constructed as it seems that the model isn't capable of learning at all doesn't.

| **Batch Size** | **Number Iterations** | **Learning Rate** | **Loss Criterion** | **Output Activation Layer** | **Accuracy** |
|---|---|---|---|---|---|
| 5 | 500 | 0.0001 | Negative log loss | Log soft max | low: 8%<br>high: 12% |
| 15 | 500 | 0.001 | Negative log loss | Log soft max | low: 6%<br>high: 14% |
| 15 | 500 | 0.01 | Negative log loss | Log soft max | low: 3%<br>high: 5% |
| 30 | 500 | 0.01 | Negative log loss | Log soft max | low: 6%<br>high: 7% |
| 50 | 500 | 0.01 | Negative log loss | Log soft max | low: 8%<br>high: 14% |
| 5 | 500 | 0.0001 | Cross entropy | Soft max | low: 8%<br>high: 21% |
| 15 | 500 | 0.001 | Cross entropy | Soft max | low: 6%<br>high: 12% |

| | | | | | |
|---|---|---|---|---|---|
| 30 | 500 | 0.01 | Cross entropy | Soft max | low: 5%<br>high: 8% |
| 50 | 500 | 0.01 | Cross entropy | Soft max | low: 6%<br>high: 8% |
| 5 | 500 | 0.001 | Multi Margin Loss | Soft max | low: 4%<br>high: 7% |
| 5 | 500 | 0.0001 | Multi Margin Loss | Soft max | low: 7%<br>high: 12% |
| 20 | 500 | 0.01 | Multi Margin Loss | Soft Max | low: 4%<br>high: 10% |
| 80 | 500 | 0.01 | Multi Margin Loss | Soft Max | low: 3%<br>high: 11% |

Consequently, exploring more deeply if this issue is with the suitability of the LSTM on our classification task and not that it is an incorrect implementation is imperative to our research. In an attempt to explore this question, we adjust the input data target to be specific to the class of video True and False, instead of the specific video and inspect the results. The expectation was that this model would perform well on the data in line with the results produced by [1]. However, this proved not to be the case. The binary classification task produced results that were a little bit above the benchmark of random guessing. This indicates even more strongly, that the sequential relationship of the pupil data is not as important a predictor for this kind of classification. It is also possible that it is an indicator of a flaw in the model and this will be explored further.

Given that there really is not enough training samples for each class, there is perhaps just not a significant enough feature representation here to accurately build the rest of the model in its current state to perform well. However, we do find that there is a high level of predictive ability on the dataset features overall so it is interesting to explore this further and consider re-analysing the dataset in this context. As discussed in the method sections, the BDNN required the data to be transformed into a one-to-one mapping of feature patterns and outputs. None of the attempted transformations produced a high level of accuracy in the determining of a representative feature pattern for the classes. It is likely that there is a high proportion of outliers in the dataset that can be skewing the way this is analysed in each of this method, or that there is not enough. It is plausible that a more interesting analysis of the features could be developed if instead of considering that each video is considered a different class. However, if we actually had the videos it is possibly that they could be manually sorted into more classes that had similar videos in each. The prediction of specific classes, is seen as a building block for further developing this model to be applied to more useful categories or classes.

## 4 Conclusion and Future Work

While there are exciting and obvious benefits to being able to effectively determine classes of videos of anger, the models investigated did not perform well here for this project did not manage to perform reliably at all. With a simple neural network the model could predict about 25% of the data, with chance being 5% and the basic neural net about 55%. The model did not produce any effective results in using an observers pupillary response to predict which video they were watching. Statistical analysis of the input features are used to generate class representations and the BDNN model is subsequently trained and fitted on this dataset. None of the implementations of this proved to be particularly effective in the grouping of the dataset and further research could definitely involve analysing if this can be optimised. An LSTM was implemented on a time-series extended version of the original dataset of the same experiment. The results were similar to random and this was close to reproduced with the same LSTM being implemented with a binary classification on the veracity of anger class. Some conclusions from this are that, disregarding the possibility of faulty code, this data could indicate interesting information regarding the nature of pupils over time and if pupil response to emotion recognition has a strong time component. Steps such as more extensive tuning and further exploration of padding techniques are necessary to rule out that the LSTM is wrong.

A first step in developing future work before expanding on this would be to explore some of the suggested techniques as suggested in the discussion section of this report to analyse the effectiveness of this technique. Further investigation involving just the videos labelled genuine was investigated, to determine if perhaps there was more significant variations within specific classes of 'Genuine' or 'Posed classification could also be useful in the models analysis.

## References


1. Chen, L., Gedeon, T., Hossain, M.Z., Caldwell, S.: Are you really angry? Detecting emotion veracity as a proposed tool for interaction In: Proceedings of the 29th Australian Conference on computer-human interaction, Brisbane Nov 28 – Dec 1 (2017)
2. Hossain, M.Z., Gedeon, T.: Classifying Posed and Real Smiles from Observers' Peripheral Physiology In: PervasiveHealth '17: Proceedings of the 11th EAI International Conference on Pervasive Computing Technologies for Healthcare, May 23-26, ACM, Barcelona vol. 1:460-463 (2017)
3. Nejad, A. F., Gedeon, T.D.: Bidirectional neural networks and class prototypes. In Neural Networks, 1995. Proceedings., IEEE International Conference on, vol. 3, pp. 1322-1327. IEEE. (1995).
4. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning internal representations by error propagation. In: Rumelhart, D.E., McClelland, *"Parallel distributed processing,* MIT Press, Vol. 1, (1986)
5. Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Comput*. 9, 8 (November 15, 1997), 1735–1780. DOI:https://doi.org/10.1162/neco.1997.9.8.1735 (1997)