Grammatical Facial Expression Recognition

Luca Di Carlo

Word Count: 2187

Introduction

Using machine learning to recognize facial expressions is a simple problem with a plethora of approaches that can be taken to achieve success in implementing a suitable algorithm. This particular problem is a binary classification that required implementing supervised learning techniques. In this paper two approaches will be considered: support vector machine classification (SVM) and a neural network (NN). Both approaches will be analyzed for performance using a multitude of parameters, including hyperparameter tuning and mitigations for poor performance will be provided where available. The analysis was completed in Python, with two Jupyter Notebooks that were produced containing code, figures, and the analysis along with comments. The notebook for the SVM is named *GFE_Classification_Training_Testing_SVM.ipynb* and for the NN it is named *GFE_Classification_Training_Testing_NN.ipynb*.

The paper will consist of this brief introduction followed by a short description of the data preprocessing. Then tasks (A) to (D) will be broken into four sections, followed by a fifth section for task (E) which will conclude this paper.

Data Preprocessing

The two facial expressions that were chosen for training and testing were negative and emphasis. Exploration of the data revealed that the data were separated into three major distributions, each for the corresponding coordinate axes (x, y, z) for a total of 300 features (100 per coordinate). Figure 1 displays the varying distributions of the data, which is neither normally distributed nor are the coordinates within the same scale. It is standard practice to transform data that is not scaled properly, and in this analysis two methods were explored: z-score (standardization) and min-max scaling (normalization). A comparison of the raw data, standardized data, and normalized data is provided in Figure 2 where it is evident that standardization transformed the data best, however this will be elaborated on more in the section of training the SVM.

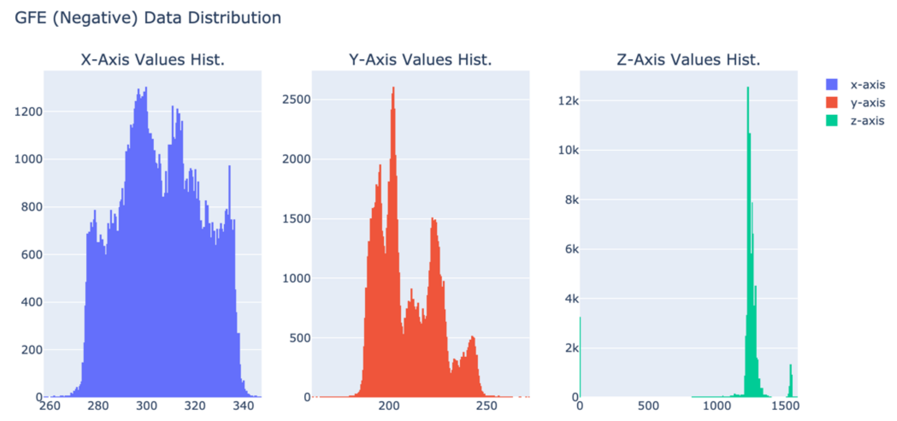


Figure 1: Histograms of the data indicate that the distributions of the data vary by coordinate axis. In addition, the scales of the data vary.

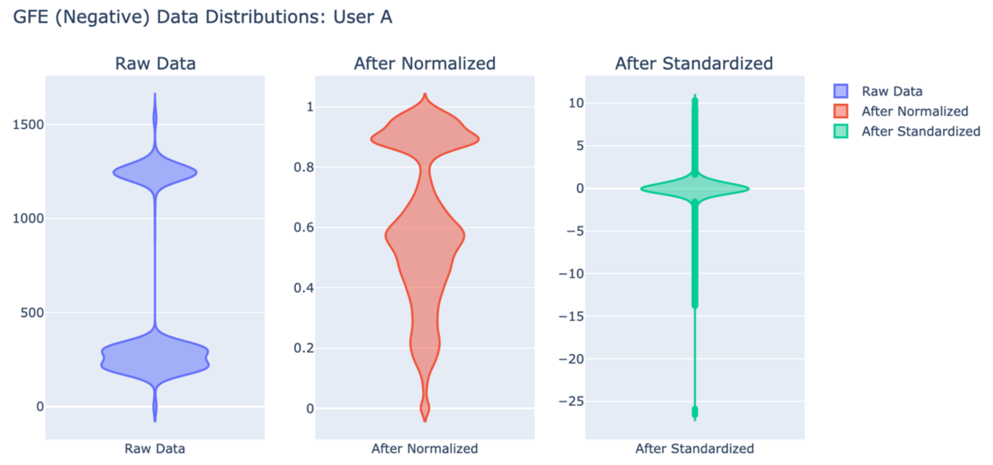


Figure 2: Plots of the distributions before and after scaling. Standardization offers the best redistribution of the data, whilst normalization does not remove the large spikes in the distribution.

Task A

Training of the classifiers were done using an 80/20 train test split from 5-fold cross validation.

Training

An SVM classifier was chosen since it is a robust classification method that is easy to implement, powerful, and is difficult to overfit. In addition, hyperparameter tuning of this classifier is simple with the major parameters of interest being the regularization parameter ("C" value) and kernel type. Training of this classifier was done first on the facial expression negative for user A, and results of the training used a Python package "gridsearchcv" to optimize the hyperparameters. In order to determine a superior method of scaling, hyperparameter tuning was performed on the raw, normalized, and standardized data and training accuracy was returned. Results in Table 1 proved that both normalization and standardization nearly identical training accuracy, however standardization was chosen as the preferred method since it appeared to visually smooth the data better in Figure 2.

Method	**Negative expression training accuracy (5-fold)**	**Emphasis expression training accuracy (5-fold)**
Raw Data	88.7 %	97.0 %
Normalization	90.6 %	97.9 %
Standardization	90.5 %	97.8 %

Table 1: Training performance by scaling technique.

The results of tuning the standardized data using 5-fold cross validation produced an SVM with a regularization value of 0.1 and a linear kernel.

This method was repeated for the facial expression emphasis, and again standardization was chosen as the preferred scaling technique based on the same reasons for negative above. Hyperparameter tuning produced a trained model with a 5-fold validation accuracy of 97.8%, and a regularization parameter of 1 and linear kernel.

Coded Implementation

A K nearest neighbour (KNN) classifier was coded from scratch as a class of functions to compare with the SVM classifier. Three distance functions were considered: Euclidean distance, Manhattan distance, and Hamming distance. Using grid search each distance function was considered along with a kernel size, and Euclidean distance with a kernel size of 15 was determined as the best performing model. 5-fold cross validation produced a training accuracy of 83.5% on facial expression negative. However due to the excessive amount of computational time (8 hours) and low accuracy, the KNN classification method was abandoned in favor of the better-performing SVM classifier.

Task B

Testing

The “negative” SVM was tested using the data from negative user B. Results indicated a poor model that could not classify negative emotions. Reasons for this could be explained by SVM using a value of 0.1 for C, which is a normal regularization value. The smaller this value, the more likely it is to misclassify points, and therefore make a more "generalized" model. Therefore re-tuning the parameters was performed in the hope that a smaller value of C would produce a more “general” model and perform better. Results on Table 2 indicate that there was a slight increase in accuracy, precision, and recall from using a smaller value of C.

Results for negative indicate that the model trained on user A is poor at recognizing user B negative emotions. Accuracy, precision, recall, and AUC all indicate that the model is slightly better at prediction than random selection. Figure 3 shows the ROC curve of testing on user B, which suggests a random model due to the flatness of the line.

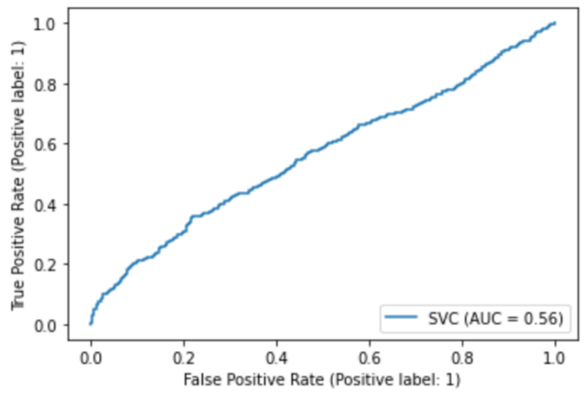


Figure 3: Very poor ROC curve for negative. Flat line indicates the model cannot distinguish between classes. Random model.

	Negative		**Emphasis**	
Performance Parameters	**Before adjusting C**	**After adjusting C**	**Before adjusting C**	**After adjusting C**
Accuracy	54.7 %	56.9 %	77.2 %	82.4 %
Precision	49.7 %	52.0 %	67.4 %	75.5 %
Recall	53.2 %	54.3 %	81.5 %	82.3 %
AUC	0.56	0.56	0.86	0.89

Table 2: Performance of training on user A and testing on user B.

The “emphasis” SVM classifier was tested using the emphasis user B data. Results indicated a decent fitting model, and in a similar fashion to negative above, the regularization parameter was adjusted and yielded a significantly better model (Table 2). Accuracy, precision, recall, and AUC improved significantly. Overall the SVM trained on emphasis user A and tested on user B produced a good model with a very good balance between specificity and sensitivity, which is evident in the bend of the ROC curve (Figure 4).

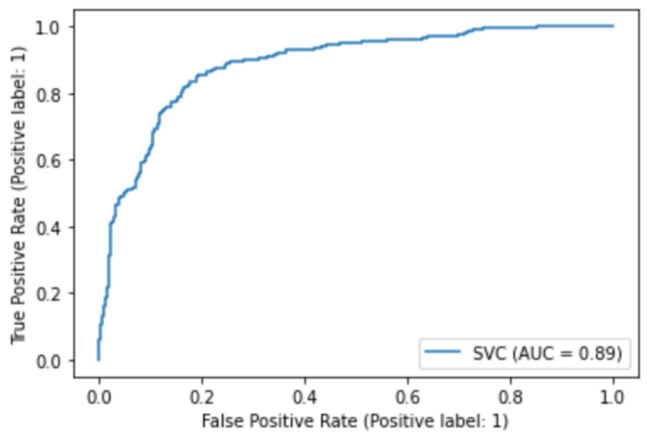


Figure 4: ROC curve of emphasis. Very good tradeoff between sensitivity and specificity of the model. Curve bends to the upper left corner.

Task C

Inverting Roles of the User

The same approach in task A and B were applied in task C, with only the role of the users being inverted and standardization as the default scaling technique. Hyperparameter tuning was performed on negative user A data and resulted in an SVM with a radial kernel and a regularization value of 0.1. 5-fold cross validation however produced an accuracy slightly greater than task A. The change in accuracy and kernel was unexpected and therefore explored by checking the distributions of the two users, however no differences in distribution were identified.
Reducing the regularization parameter resulted in a significant improvement of performance greater than that observed in task A, although model performance remained mediocre (Table 3). Model accuracy, precision, and recall were mediocre whilst the ROC curve indicates a model that is able to distinguish between classes well.

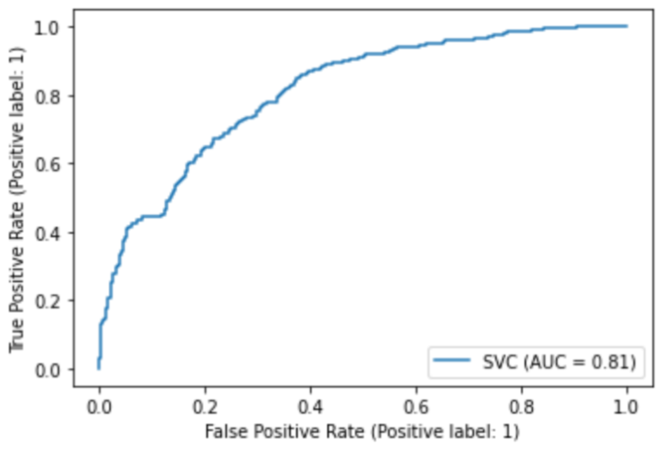


Figure 5: ROC curve of negative using reversed roles. ROC curve indicates a decent model, slight curve to upper left.

	Negative		**Emphasis**	
Performance Parameters	**Before adjusting C**	**After adjusting C**	**Before adjusting C**	**After adjusting C**
Accuracy	60.7 %	72.2 %	84.6 %	85.2 %
Precision	61.2 %	74.9 %	73.8 %	77.6 %
Recall	43.1 %	61.2 %	53.6 %	52.4 %
AUC	0.68	0.81	0.66	0.85

Table 3: Table of performance after reversing roles.

For training emphasis user B the optimized parameters were C=0.001 and a linear kernel. Re-adjustment of the regularization parameter yielded a slightly better model, with a significant increase in AUC (Table 3). Recall however did not improve and suggests the model correctly identifies emphasis only 52.4 % of the time, regardless of the good ROC curve (Figure 6).

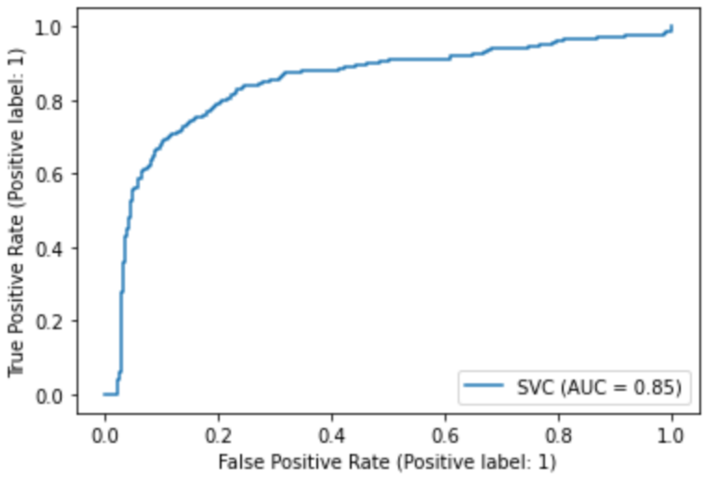


Figure 6: ROC curve of emphasis tested on user B, curve extends to top left corner which suggests good model specificity and sensitivity.

Different Feature Representation

Principle component analysis (PCA) was chosen to perform dimensionality reduction on both facial expressions. A scree plot of negative user A features indicates that 23 components comprise over 99% of the variance (Figure 7). Training the SVM on user A and testing on user B with only 23 components was performed and the results in Table 4 indicate that little change was observed in performance. Results of PCA on negative user B produced 81 components, which is almost four times more than user A. This suggests that between users there are varying numbers of important features and may explain the differences in training and testing model performance between users of the same facial expression. PCA on emphasis user A resulted in 22 components comprising of 99% of the variance (Figure 7). Training on user A and testing on user B resulted in a very small decrease in model performance, however a good model nonetheless.

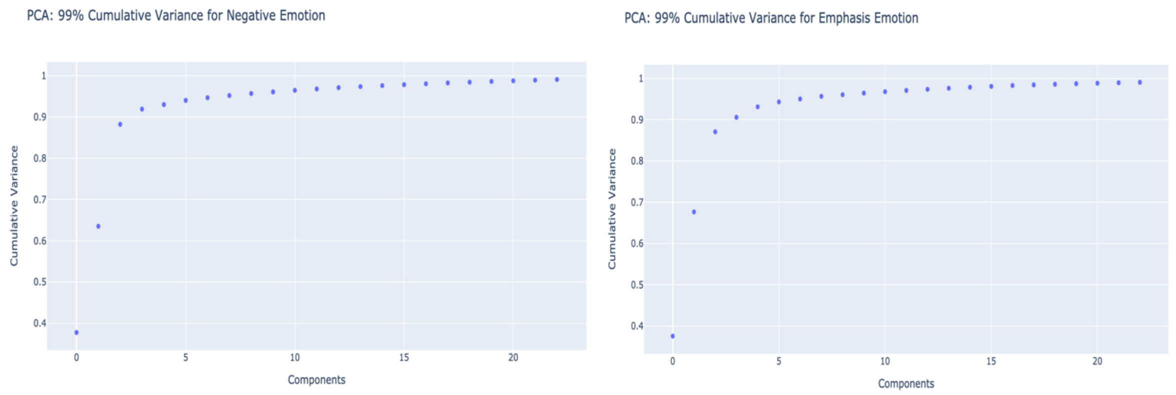


Figure 7: Plot of cumulative variances for expressions. 22 and 23 components respectively.

	Negative		**Emphasis**	
Performance Parameters	**Before PCA**	**After PCA**	**Before PCA**	**After PCA**
Accuracy	56.9 %	59.6 %	82.4 %	81.4 %
Precision	52.0 %	54.8 %	75.5 %	75.1 %
Recall	54.3 %	58.0 %	82.3 %	79.1 %
AUC	0.56	0.62	0.89	0.88

Table 4: Table of PCA performance.

Conclusions on task C

Inverting the training and testing roles resulted in slight changes to model performance. PCA slightly improved negative expression recognition which was unexpected, however this improvement still resulted in a very poor model. Emphasis saw a slight reduction in performance after PCA. Dimensionality reduction substantially reduced the number of features from 300 to 22 and 23 respectively, whilst not severely impacting model performance.

Task D

Training and Testing

A NN was chosen as the second classification method after partial inspiration from a paper by Wallawalkar (2017) which explored an architecture using the same grammatical facial expression dataset. Wallawalkar (2017) developed a network which reduced the number of neurons in each layer to learn specific patterns in the data. This was applied to the architecture in Figure 8, with 300 neurons in layer 1 for each coordinate axis feature, 100 neurons in layer 2 for the recognition of the coordinate facial structure, 10 neurons in layer 3 for the recognition of facial attributes (nose, mouth, etc), and a two-class binary output since it produced the best results. Hyperparameter tuning was performed using grid search and 5-fold cross validation, which chose “relu” as the activation function and dropout rate of 0.3.

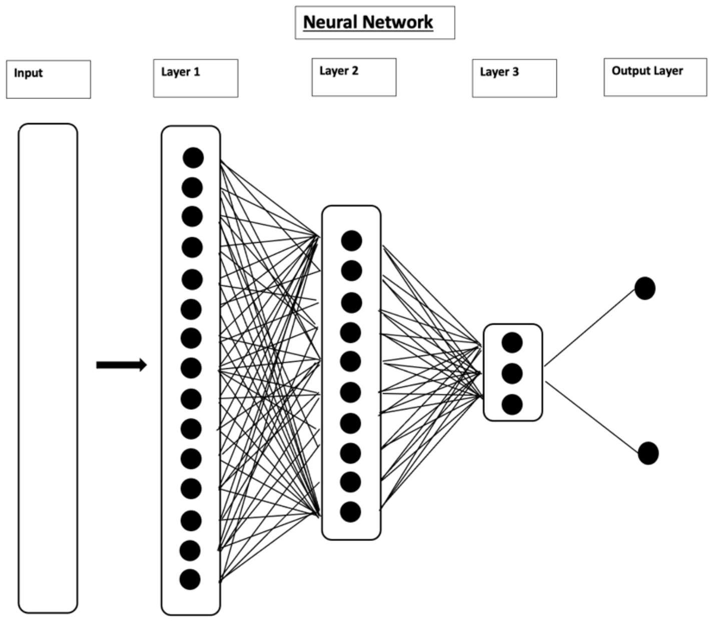
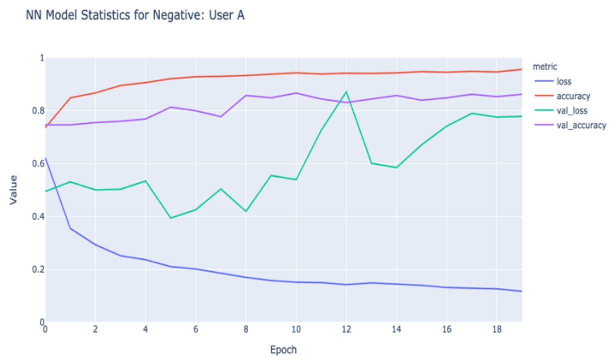


Figure 8: NN architecture.

The network was trained on negative user A, using only 5 epochs and a batch size of 100 to avoid overfitting (Figure 9). Testing on user B negative resulted in a poor performing model (Table 5) that is slightly better than random selection. Repeating the training on emphasis user A resulted in using 10 epochs to avoid overfitting and very good model performance, boasting 87.3 % accuracy and an AUC of 0.87.



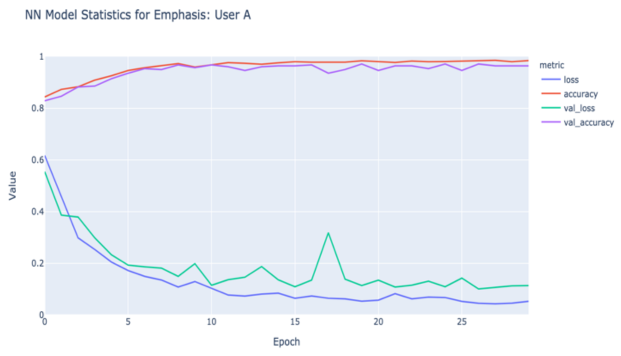


Figure 9: Performance of the NN for each expression. Looking to minimize loss and validation loss. Once validation loss begins increasing and validation accuracy flatlines, model begins overfitting.

Performance Parameters	Trained on user A tested on user B		Trained on user B tested on user A	
	Negative	**Emphasis**	**Negative**	**Emphasis**
Accuracy	60.8 %	87.3 %	55.6 %	71.6 %
Precision	55.5 %	82.3 %	51.9 %	40.6 %
Recall	57.7 %	86.3 %	76.5 %	44.5 %
AUC	0.60	0.87	0.56	0.62

Table 5: NN performance. NN architecture can be applied to different facial expressions, however reversal of the users results in poor performance.

Inverting the roles of the training and testing users resulted in poor results for both negative and emphasis expressions. Unlike in the SVM, training on user B and testing on user A for emphasis resulted in a very poor model which had low recall and precision, therefore unable to distinguish between positive and negative class labels.

Different Feature Representation

Again suggested by Wallawalkar (2017), a different feature representation was proposed in his paper which involved dividing the inputs by facial feature before feeding them into the network. This was adapted and implemented (Figure 10). 10 inputs correspond to the 10 facial attributes with each having a different shape based on the number of coordinate points. The reasoning for this approach is to pre-define the facial features of the network, which when the training and testing roles are inverted will result in better model performance.

Each input is two-dimensional, with the extra dimension holding the coordinate axes (x,y,z). Each input is followed by a layer which reduces the facial attribute coordinates to one neuron, which allows the network to recognize each facial attribute. Each of the 10 layers return one neuron, which are concatenated together into the second layer which contains a 10 neurons of (1,3) shape, or altogether a (10,3) shape. A dropout layer and batch normalization are applied to delay overfitting in training, and the final layer is flattened to a 30 neurons and output through a 'softmax' function to produce a binary two class label output. Hyperparameters from the previous network were applied to this network, using a dropout rate of 0.3 and "relu" activation function.

Results from training and testing on both emotions, in addition to inverting the roles of the users are displayed in Table 6.

Performance Parameters	**Trained on user A tested on user B**		**Trained on user B tested on user A**	
	Negative	**Emphasis**	**Negative**	**Emphasis**
Accuracy	62.9 %	85.6 %	59.1 %	72.4 %
Precision	60.7 %	87.0 %	58.4 %	41.1 %
Recall	50.0 %	74.6 %	44.5 %	40.0 %
AUC	0.62	0.84	0.58	0.61

Table 6: Table of multi-NN performance. No significant improvement from original NN.

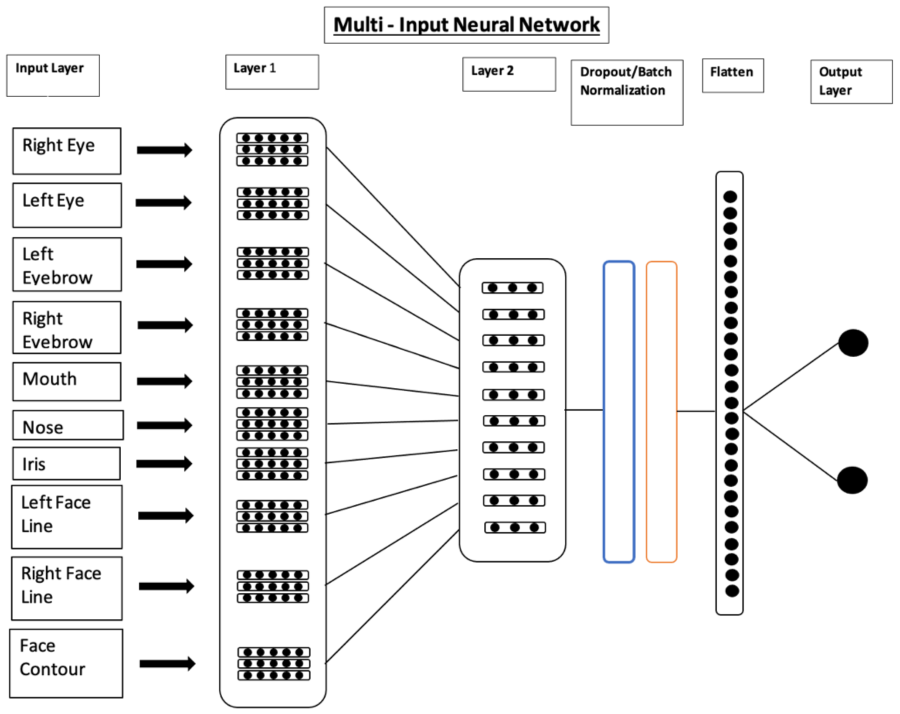


Figure 10: Multiple input neural network architecture. Input layer and layer 1 are divided among the 10 facial attributes. Only in layer 2 is the network concatenated and combines the 10 facial attributes.

Task E

Conclusion

Both approaches were able to identify emphasis expressions with greater accuracy than negative (Table 7). This is synchronous to the paper by Freitas *et al.* (2014) which saw good model performance on assertion (emphasis) and poor performance on negative. The NN produced slightly better results for both expressions when training on user A and testing on user B. In addition, the NN required a single architecture that performed well on both expressions. However, inverting the roles of the user resulted in very poor model performance by the NN. SVM was far better in performance after inverting the users, and this may be because using hyperplane to separate data is a more generalized method of classification between users.

The multiple-input NN (multi-NN) used for different feature representation failed to improve the NN when inverting the users. This suggests that in order for model performance to maintained, hyperparameter tuning or even model restructuring may be necessary when training on user B and testing on user A. The multi-NN explored a different architecture that may be elaborated on in further testing. Perhaps using more layers in reducing each of the 10 input layers could improve model performance or increasing the number of layers within the concatenated network.

	Trained on User A tested on User B				**Trained on User B tested on User A**			
Performance Statistic	**SVM**	**PCA SVM**	**NN**	**Multi-NN**	**SVM**	**PCA SVM**	**NN**	**Multi-NN**
Negative								
Accuracy	56.9 %	59.6 %	60.8 %	62.9 %	72.2 %		55.6 %	59.1 %
Precision	52.0 %	54.8 %	55.5 %	60.7 %	74.9 %		51.9 %	58.4 %
Recall	54.3 %	58.0 %	57.7 %	50.0 %	61.2 %		76.5 %	44.5 %
AUC	0.56	0.62	0.60	0.62	0.81		0.56	0.58
Emphasis								
Accuracy	82.4 %	81.4 %	87.3 %	85.6 %	85.2 %		71.6 %	72.4 %
Precision	75.5 %	75.1 %	82.3 %	87.0 %	77.6 %		40.6 %	41.1 %
Recall	82.3 %	79.1 %	86.3 %	74.6 %	52.4 %		44.5 %	40.0 %
AUC	0.89	0.88	0.87	0.84	0.85		0.62	0.61

Table 7: Complete table of performances across all modelling approaches.

SVM Optimized Hyperparameters				
	Trained on User A tested on User B		**Trained on User B tested on User A**	
Hyperparameter	**Negative**	**Emphasis**	**Negative**	**Emphasis**
Kernel	linear	linear	radial	linear
C	0.01	0.01	1	0.0001
Degree	n/a	n/a	n/a	n/a

Table 8: Table of optimized parameters for SVM.

References

Walawalkar, D. (2017) 'Grammatical facial expression recognition using customized deep neural network architecture'.

Freitas, F.*, et al.* (2014) 'Grammatical facial expressions recognition with machine learning', pp. 180-185.