Machine Learning Capstone:
YouTube Facial Recognition Using Landmarks

Nicole Woodland, July 2019

Phase 0: Data Acquisition

This dataset is a processed version of the YouTube Faces Dataset, that contained short videos of celebrities that are publicly available and downloaded from YouTube. There are multiple videos of each celebrity (up to 6 videos per celebrity). The original videos were cropped around the faces and only 240 consecutive frames were kept for each original video.

The landmarks were created using the algorithm created by 2D and 3D keypoints, Adrian Bulat and Georgios Tzimiropoulos as documented in 'How far are we from solving the 2D & 3D Face Alignment problem?' (and a dataset of 230,000 3D facial landmarks), arxiv, 2017. (pdf)

The YouTube dataset used is available here: https://www.kaggle.com/selfishgene/youtube-faces-with-facial-keypoints

Phase 1: Data Analysis and Preparation

In the large dataset, there are 374 individuals with a total of 155,560 individual frames. The images are not of equal dimensions, have varying colour tone and quality, have all angles of facial orientation and uncentered faces making normal convolutional networks difficult without substantially more processing. Using the landmarks allows for faster and more consistent predictions.

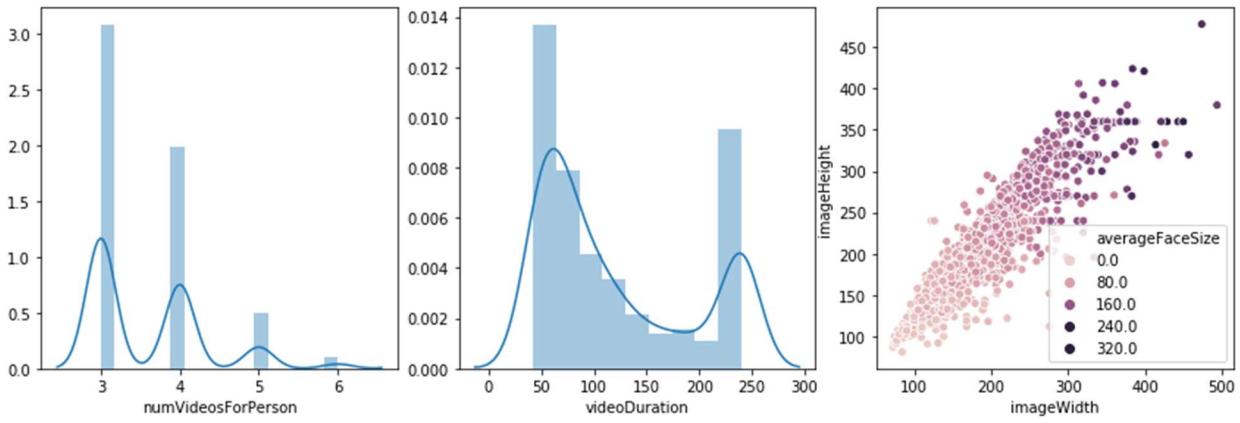

Figure 1: Left-Distribution plot of the number of videos per person available, Centre- Distribution of the length of the videos (measured in number of frames) and Right - Distribution of the image dimensions.

Each .npz file contains four columns:

1. ['colorImages'] containing arrays of the image colour values
2. ['boundingBox'] containing the rectangle hand applied to the image to locate the face
3. ['landmarks2D'] containing 68 sets of (x, y) coordinates representing landmark locations
4. ['landmarks3D'] containing 68 sets of (x, y, z) coordinates representing landmark locations

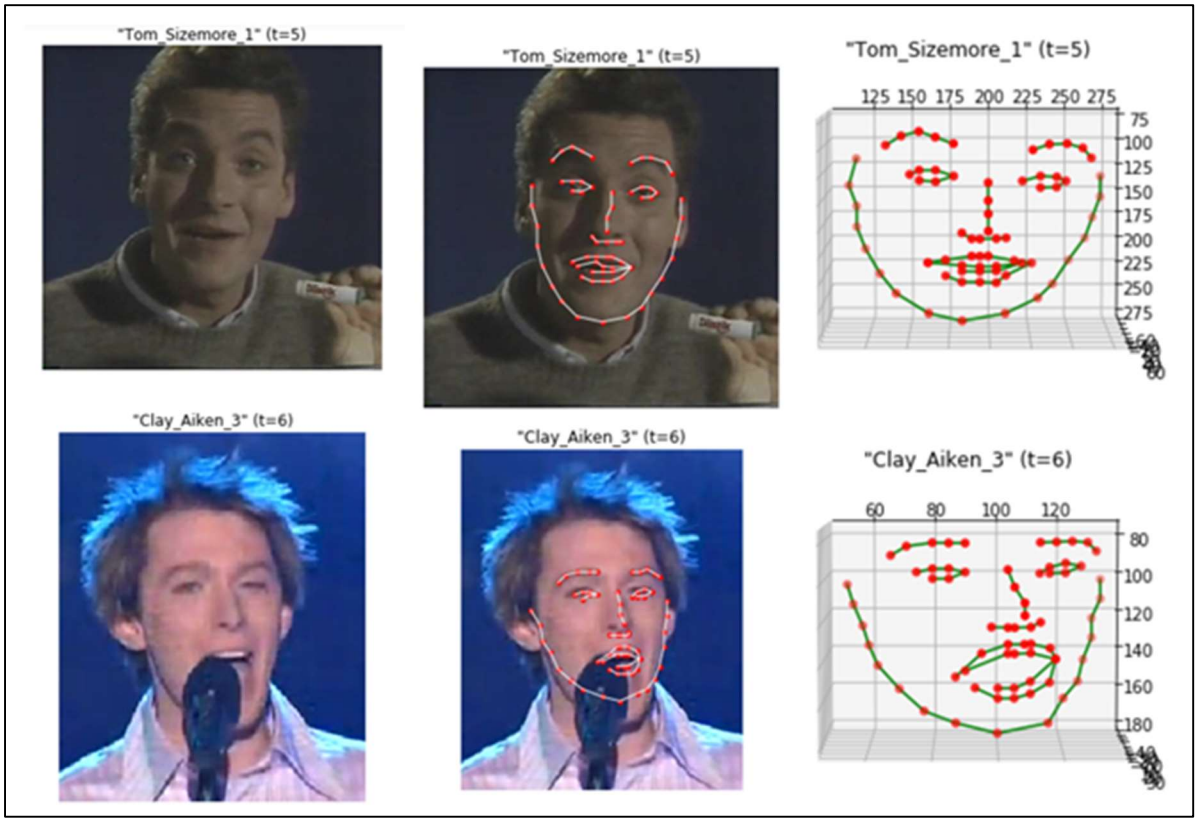

Figure 2: A sample of the image, 2D landmarks and 3D landmarks provided in the dataset

Because the landmarks are located wherever the face is in the image, we need to normalize and standardize the coordinates before they can be used in machine learning algorithms. A method to normalize the landmarks was included in the Kaggle database, and I have used it here with success.

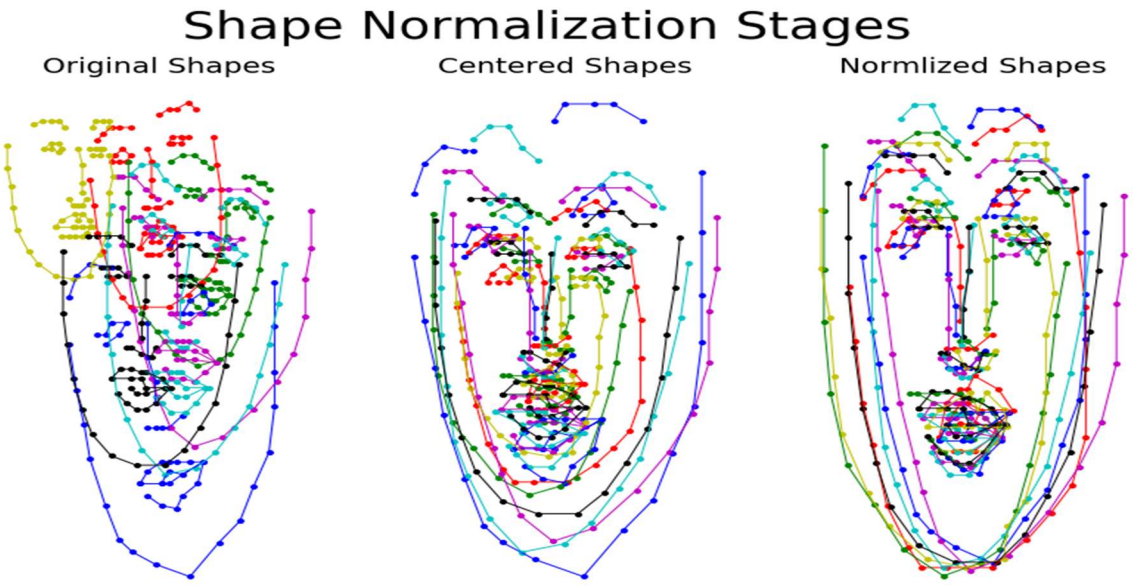

Figure 3: Normalization stages of the facial landmarks for a few frames in the database. Faces are collected, centered and fit to a set bound.

Phase 2: Model Selection

The rationale behind this project is that a photographer wishes to develop a mobile app that would allow them to take a photo of a client at the time of consultation which is added to the database. When the photographer later uploads photos of the client, the ap would recognize the client and send them an automatic notification and link that their images are available for viewing and download. This would require an easily and quickly updatable machine learning model to allow the input of new people and the ability to match a client with their contact information based solely on facial recognition.

This is a **classification problem**.

We want to identify a unique pattern of landmarks as a specific person.

Because the landmarks come in coordinate pairs and are spatially significant, I focused the majority of my effort on optimizing a simple neural network and also evaluated clustering and classification models.

Models Used:

1. Clustering
2. Logistic Regression (LR)
3. Decision Tree Analysis (DT)
4. K-Mean Nearest Neighbours (KNN)
5. Support Vector Classification (SVC)
6. Gaussian Naive Bayes (GNB)
7. RandomForestClassifier (RFC)
8. Forward Propagating Multi-Layer Perceptron Neural Networks

Phase 3: Model Design and Parameter Selection

The parameters chosen to optimize each model are available within the associated Jupyter notebooks. A sample is provided here including the final chosen models.

Clustering

Clustering was optimized using the elbow method. A plot of the sum of squared error (SSE) vs the number of neighbours shows the optimum number of clusters around the inflection point. I chose 6 in this case as well as the number of people in the dataset, as theoretically there should be a cluster for each person.

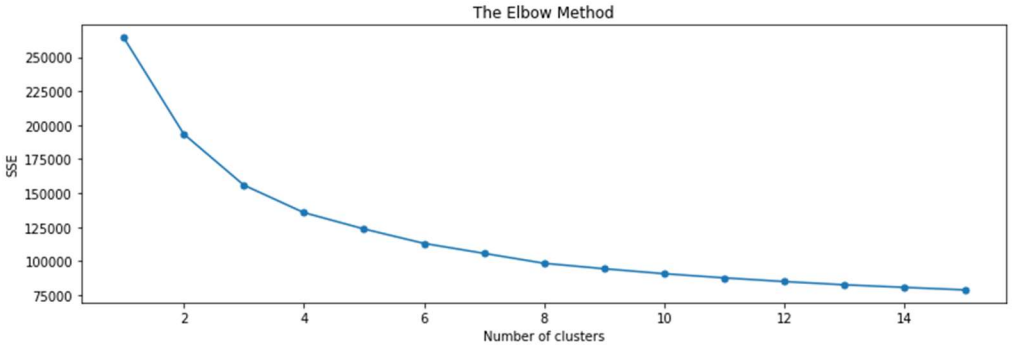

Figure 4: Elbow method for optimizing clustering

Classification

Default values were used in classification methods as the initial network I tried immediately had over 90% accuracies. Because of the loss of spatial relationship of the coordinates in the way that the data is forced to be run through the classification methods, I didn't expect to see very high results and I opted to settle for the defaults initially.

Neural Networks

The Initial networks were set-up with few layers, no dropout and no early stopping. The input layers have either 136 or 204 inputs for the 2D and 3D landmark sets respectively. The output layer has either 27 or 374 nodes depending on the number of people in the dataset, 27 for the subset, and 374 for the entire thing. After reviewing the behaviour of the log-loss plots, I would adjust the number of layers, the number of nodes, the batch size and the number and location of drop-out layers to try and reduce overfitting and increase the accuracy of the model.

The 2D network was quite simple and extremely effective.

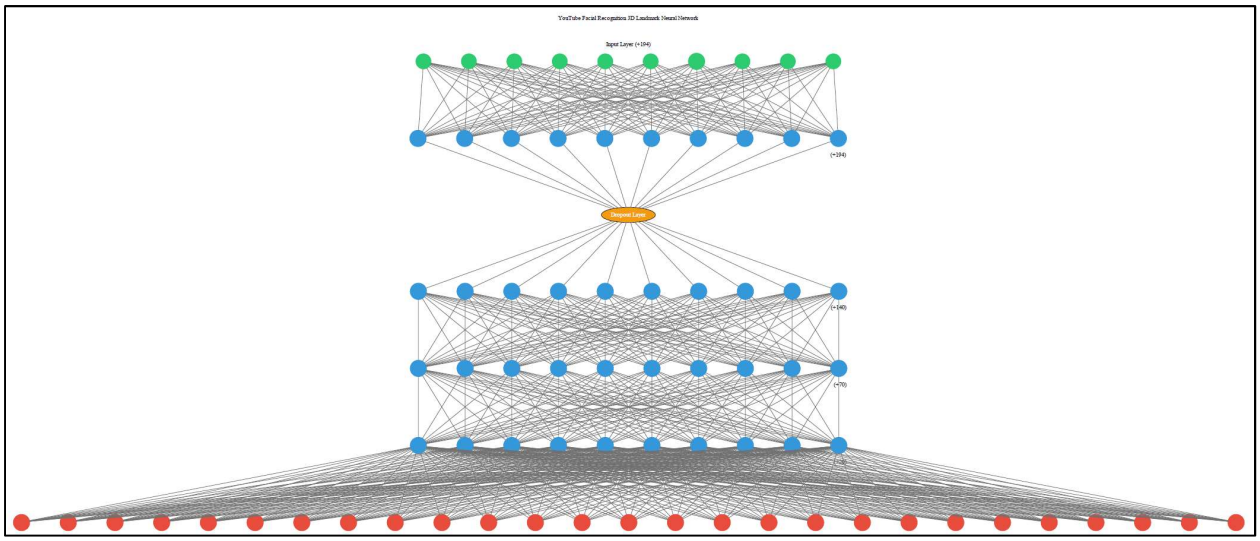

Figure 5: Visual representation of the neural network for 27 outputs containing 4 layers, 1 dropout layer and it ran with a batch size of 20 for 300 epochs.

The 3D network for 374 individuals was substantially larger with a total of 731,684 parameters. Two drop-out layers were eventually required to prevent overfitting, and the upscaling and downscaling of the number of internal nodes took some patience and pattern tracking to optimize the network. I believe the network could still easily improved, but it become a time vs improvement question as the number of people in the database increase. For the case of our

Layer (type)	Output Shape	Param #
dense_104 (Dense)	(None, 204)	41820
dense_105 (Dense)	(None, 400)	82000
dropout_18 (Dropout)	(None, 400)	0
dense_106 (Dense)	(None, 400)	160400
dropout_19 (Dropout)	(None, 400)	0
dense_107 (Dense)	(None, 390)	156390
dense_108 (Dense)	(None, 380)	148580
dense_109 (Dense)	(None, 374)	142494

Total params: 731,684
Trainable params: 731,684
Non-trainable params: 0

Figure 6: Internal structure of improved neural network

photographer, it might work quite well for a long time before they exceed the reliability of the network with hundreds of clients.

Phase 4: Model Training and Evaluation

The trained models can all be found documented within the accompanying Jupyter Notebooks. Here is a summary of my thoughts on each of the models:

1.0 Clustering

Clustering does not work at all in this way. The clustering favours the orientation of the face over the spatial relationships and therefore groups images by direction the face is looking, not by who is looking.

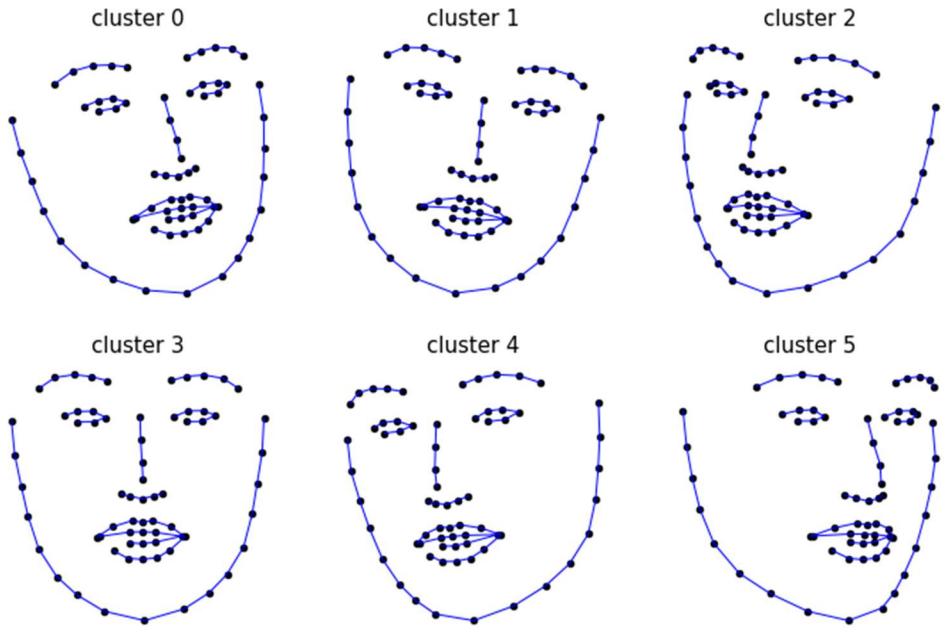

Figure 7: Sample of landmarks from the clustering process. It favours orientation of the face over the uniqueness.

A sample of one of the clusters from the big dataset. The method is so poor for recognition, it only managed a 1-2% accuracy. Nearly every person is in each cluster.

Figure 8: Left - pie plot showing the number of people represented in this cluster that should be representing Alicia Silverstone (centre). A random guess from the cluster, surprise, surprise, (left) isn't a picture of Alicia.

2.0 Logistic Regression (LR)

This method worked alright for few number of individuals but would not compute for the large dataset. I wouldn't use this method going forward.

3.0 Decision Tree Analysis (DT)

Relatively sable for both small and larger datasets with reasonable accuracy. Could be further optimized.

4.0 K-Mean Nearest Neighbours (KNN)

Relatively stable and quick for both set sizes. Has accuracy comparable to the neural network for the larger dataset. The optimized number of neighbours is 1, making it a quick method to run. It could be tested for continued use. Figure 9 has the Elbow Plot showing 1 equal to the optimum. This is similar behaviour to the large dataset.

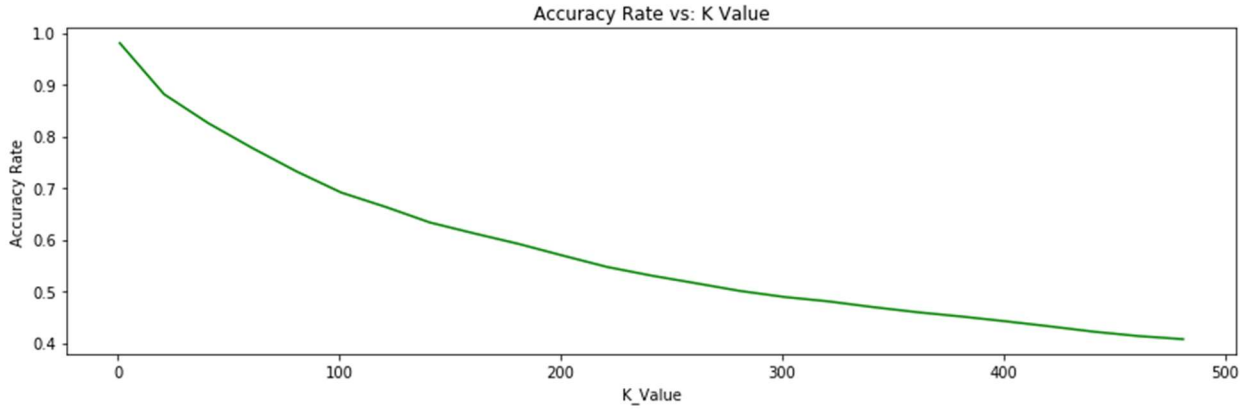

Figure 9: Elbow plot for the K-Means Nearest Neighbours method. The optimum point is where the accuracy is the highest or there's an inflection.

5.0 Support Vector Classification (SVC)

Poor response for the small dataset and time-consuming for the bigger set. Did not pursue further optimization even though it should have been a useful method as it's the most similar to a neural network. This dataset is reaching the upper range of its effective capacity though which occurs around 200,000 occurrences.

6.0 Gaussian Naive Bayes (GNB)

Stable performance for both sizes, very poor results, so I did not continue optimization.

7.0 RandomForestClassifier (RFC)

This method was expected to perform better than it did. It was stable in both datasets, and still performed alright in the larger set without much fuss. With more time, I will attempt to optimize this method.

8.0 Forward Propagating Multi-Layer Perceptron Neural Networks

The neural networks by far had the best performance of all the methods (expect perhaps the KNN), in fact the small dataset had nearly perfect accuracy. It did take quite a bit of trial and error to optimize the networks to the level they are at. They do take some time to evaluate. Like the plot for the small set model, there is minimal overfitting and a very high accuracy is reached quickly. It actually appears that the epochs could be increased and there could be even slight additional improvement.

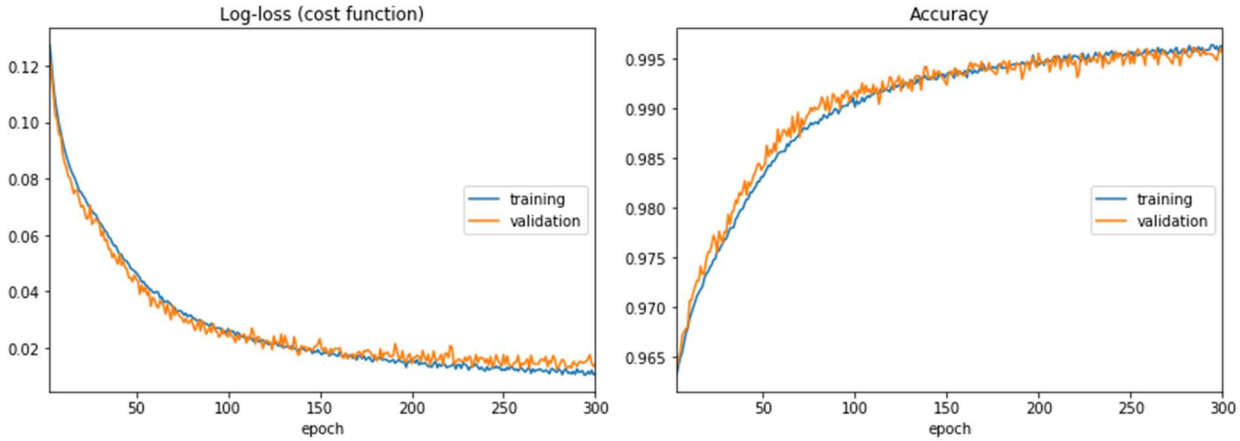

Figure 10: Neural network results for the simple network described in Figure 5, with final accuracy of 99.6%.

For the large data set, it was more difficult to find a stable model that wasn't overfitting the data. The model summary in Figure 6 resulted in the following plot which is quite reasonable. The best accuracy was 88.6%, 2.7% better than use the 2D landmarks for this dataset. Batch size was larger at 110.

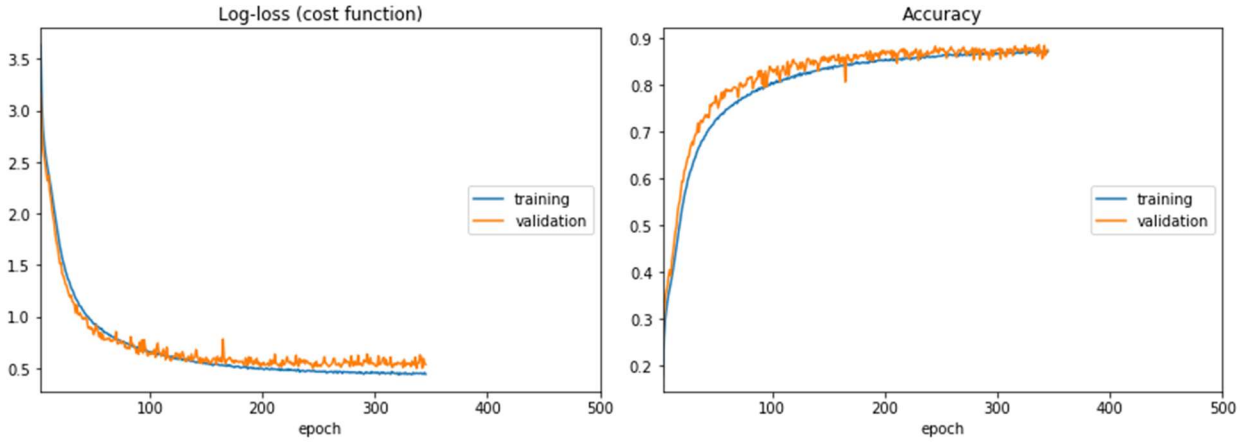

Figure 11: Log loss plot for the final 3D landmark model for the large dataset.

Phase 5: Final Performance Results and Conclusions

A comparison of all the models can be found in Figures 12 and 13. The neural networks are the most effective at teasing out the spatial relationships between the data points and allowing an image to be

matched with the proper name. The KNN method deserves additional attention and could play a role increasing certainty of a prediction by applying both methods.

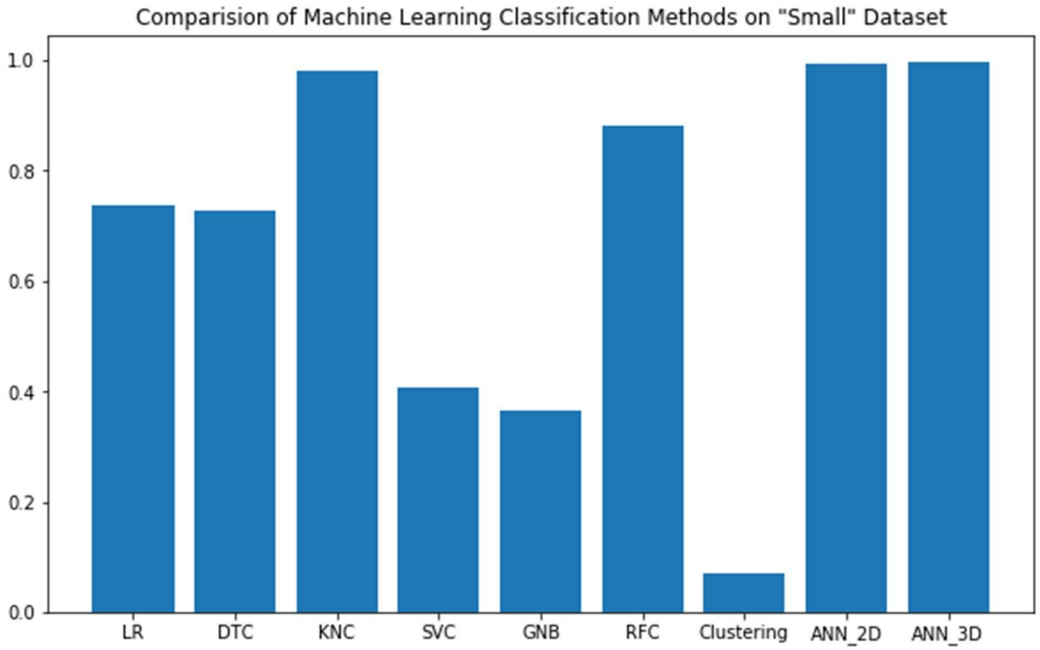

Figure 12:Comparison of accuracy scores from all methods to allow facial recognition from a photograph with facial landmarks (small dataset)

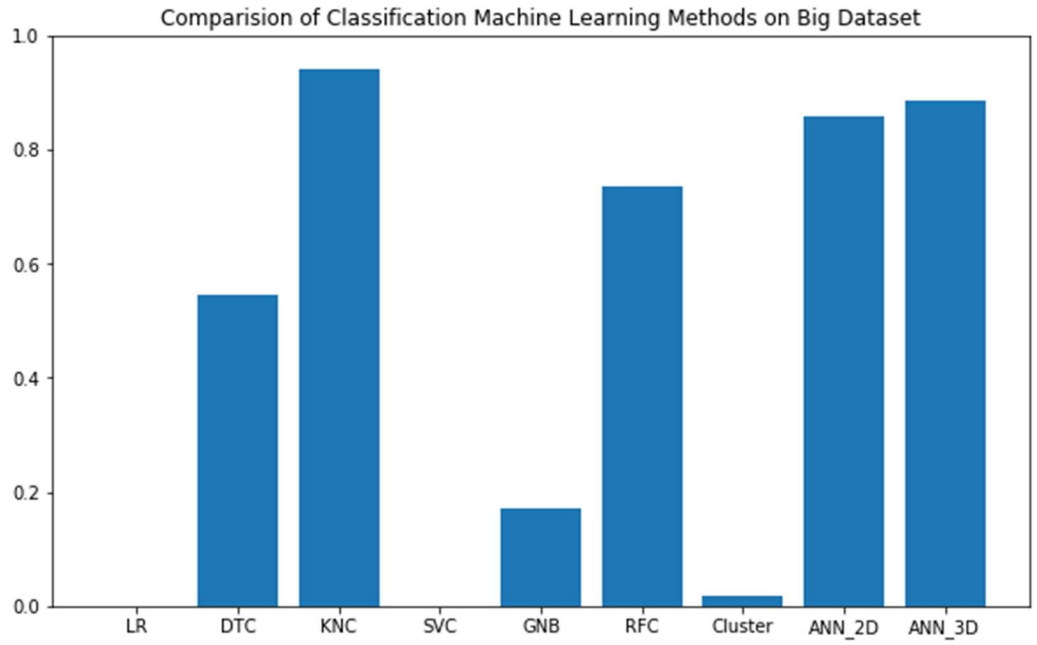

Figure 13:Comparison of accuracy scores from all methods to allow facial recognition from a photograph (full dataset). I was unable to run Linear Regression and SVC on the large dataset in time for completion of this report.

In summary, using a landmarking algorithm and machine learning can predict a person from a list with relatively good accuracy. The smaller the dataset, the better the accuracy using this method though, so something would need to be implemented so clients with more models don't have dropping accuracy.

An app built to notify a client when their photo is uploaded is possible, but would require inputting more than one image to retrain for this model. A couple short videos spanning multiple angles could work! This would give enough additional frames and different angles to well-train a model.

I would highly recommend a 'confirm' button before notification to the client from the app as the method is clearly not foolproof. This could lead to reinforcement learning in the future as the new image would be labelled in order to send out a notification.