APPLIED MACHINE LEARNING SYSTEM ELEC0134 22/23 REPORT

SN: 19076187

ABSTRACT

In this report, we evaluated several machine learning models for facial recognition tasks, including gender detection, smile detection, face shape classification, and eye colour classification. The models tested included Random Forests, Support Vector Machines (SVMs) in linear, polynomial and Radial Bias Function (RBF) kernels, and an implementation of the VGG16 Convolution Neural Network (CNN) architecture

Overall, the results showed that CNNs performed the best, achieving an accuracy of 93.8% for gender detection and 84.9% for eye colour classification and 100% accuracy on face shape classification. Random Forests performed the worst, with an accuracy of 86.5% for face shape classification and 72.1% for eye colour classification. SVM models performed similarly, with Linear SVM achieving the highest accuracy of 90.4% for smile detection. In conclusion, CNNs are a strong choice for facial recognition tasks .

1. INTRODUCTION

This report discusses machine learning solutions for the tasks of binary classification A: gender and smiling detection, and multi-class detection B: face shape and eye colour, containing 5 classes each.

For task A, the CelebFaces Attributes Dataset (CelebA) [1] is provided with 5000 training and 1000 test headshot images at various angles. For both gender and smiling detection tasks three separate models are tested. The random forests model and Support Vector Machine (SVM) models use a pretrained dlib facial detector model to extract landmark coordinates to be used as features as seen on figure 1. Three different support vector machine (SVM) models: linear, polynomial, and radial basis function (RBF) kernels, which are hyperparameter tuned and evaluated using classification metrics and performance visualization. An implementation of the VGG16 Convolutional Neural Network [2] is also implemented using the entire image as data.

Task B uses Google Cartoon Set, which contains 10,000 training and 2500 test images [3]. For face shape and eye colour detection both aforementioned preprocessing method is applied to a random forests model. Both tasks will also employ a VGG16 model for the multiclassification problems of face shape and eye colour classification.

The report is organized as follows: Section 2 provides a literary review on potential approaches to binary and multiclass classification. Section 3 gives a description of the models and the reasoning behind their selection. Section 4 presents the implementation of the of the aforementioned machine learning models used for the classification tasks. Section 4 discusses the results of the experiments and the performance of the models. Finally, in section 5, we present the conclusion and future work.

Figure 1 - 68 Facial Landmarks

2. LITERATURE SURVEY

In this literature survey we will discuss various machine learning methods used for binary and multiclass image classification. focusing on facial recognition.

Decision trees are a classical supervised machine learning algorithm that have applications in both classification and regression tasks. The algorithm were first introduced in influential paper The Iterative Dichotomiser [4]. The algorithm makes predictions by traversing the tree from the root the tree to a leaf node, where each internal node represents a test on an attribute and each leaf node represents a class label. The goal of the decision tree is to find the most informative attributes and split data at each internal node and recursively repeat this process until all the data belongs to the same class or a stopping criteria is reached. Decision trees are robust to outliers and missing data however can be sensitive to small changes, ineffective with high dimensional data and lead to prone to overfitting, particularly as the tree grows

The code is provided on GitHub: https://github.com/yshv/ELEC0134.git

larger. Therefore, would not be suitable for either of the binary or multiclass tasks due to the high dimensional data facial feature data or due to overfitting with the multiclass problems would be prone to as the trees get larger and more complex.

To address these limitations random forests, introduced in 2001 by Random forests [5] which combines multiple decision trees, where each tree was trained on a different subset of data and results are combined for a final prediction. Random forests improve upon decision trees through the reduction of variance and therefore subsequently overfitting and inability to handle high dimensional data, as decision average out predictions across multiple trees which are trained on different subsets of the dataset. Random forests have applied for facial image classification tasks [6]and shown to achieved an accuracy of 93% on the Labelled Faces in the Wild dataset (LFW) and was shown to be most robust to lighting conditions and head pose as compared to principal component analysis (PCA) and linear discriminant analysis (LDA) which are both linear methods which struggle with the curse of dimensionality, meaning that as number of features or dimensions increases the data required to model the problem increases exponentially [7]. However, random forests are sensitive to class imbalances, meaning they perform worse when there is significantly more data in one class than other classes however that this can be compensated with methods such as oversampling, under sampling and cost sensitive learning. [8]

Support vector machines (SVMs) are a type of supervised learning algorithm which was first introduced by in 1995 [9] and can be used for classification and regression tasks. SVM's work by finding the optimal hyperplane to separate data into different classes. The best hyperplane is the one that maximizes the margin, which refers to the distance between the hyperplane the closest data points from each class, which are known as support vectors. SVMs are able to handle non-linearly separable data by using kernel functions which map the original data to higher-dimensional space where the data becomes linearly separable. There are several kernels with their own strengths and weaknesses, mainly that linear kernels are applicable to linearly separable data [10]and polynomial, radial basis function and sigmoid kernel are suitable for non-linearly separable data, however are sensitive to their respective parameters and the scale of the data. SVMs also have built in mechanisms to negate class imbalance, are less sensitive to the curve of dimensionality and are efficient in terms of computational time when using kernel functions

Convolutional Neural Networks (CNNs) are a form of deep learning algorithm, where deep learning refers to a type of machine learning which uses multiple layers of processing to progressively extract higher level features from data. CNNs use convolutional layers which are designed to automatically and adaptive learn the spatial hierarchies of features from the input data. CNN's are composed of multiple layers of artificial neurons which are organised in a hierarchical structure, where the layers are organised in three parts: input layer, hidden layers and output layer. The input layer receives the raw data, in this case an image, and the output layer produces the final prediction. The hidden layers are responsible for extraction of features from the raw input data and transforming it into a representation which can be used by the output layer. The primary building blocks of CNN's is the hidden layer, also known as the convolutional layer, which extracts features using filters. These filters are responsible for detecting specific types of features such as edges or textures and are moved across the input data to compute the dot product of the filter and that specific portion of the data. This process is repeated for all the filters and the result is a set of feature maps which represent the presence of different features within the input data. Following the convolutional layers, the extracted features are passed through as pooling layer which reduces spatial dimensions of the feature map whilst maintaining the most important information, this reduces the computational cost and makes feature more robust to small translations and distortions within the input data. Finally, the features are passed through a fully connected layer (output layer) which makes final predictions. This layer is composed of a set of artificial neurons which are connected to all of the previous layer's neurons. This fully connected layer is trained to recognize complex patterns in the features to produce the final prediction.

3. DESCRIPTION OF MODELS

This section will include 2 sections: Data exploration, which will explore the dataset and discuss their characteristics and Model selection which will detail which models were selected and why based on the data exploration and give a brief overview of model itself.

Data Exploration

The selection of an appropriate model for it's given dataset is very important to achieve optimal performance. Therefore, exploration of the dataset is necessary in order to select optimal model selection. This exploration is based on such size, complexity, feature extraction, label distribution, dimensionality, and linearity of the dataset.

The CelebA dataset, while containing a relatively small number of images, is complex in nature due to the presence of real faces at various angles, lighting and pose. With a training set of only 5000 images, the risk of overfitting is high and it may not be sufficient for training and validation on more complex models. The label distribution within the dataset is balanced with 2500 images for both gender and smiling classes. Additionally, the dimensionality of these images is high due to their complexity, therefore feature

extraction may be considered as a possible approach. It must be noted extracted facial landmark features or raw images are not linearly separable. Therefore, it would be beneficial to test both simpler and more complex models and compare performance.

Whereas, the Google Cartoon dataset much larger and less complex in comparison to the CelebA dataset. The dataset consists of simple cartoon images taken from the same angle, with little different characteristics per image relative to CelebA. The label distribution is equal for face shape and roughly equal for eye colour with 2000+-30 images in each class. The dimensionality of this dataset is lower than that of the CelebA dataset due to its relative simplicity. However, it must be noted that a significant proportion of the cartoons are wearing glasses which obscure their eye colour, which can be considered noisy data. Therefore, the performance of any model which does not deal with this data will be reduced by the roughly the percentage of noisy data in the dataset. The extracted facial features or raw image data are again not linearly separable. Due to these characteristics, a more complex model with more parameters may be employed to achieve optimal performance.

Model Selection

For task A, gender and smile detection, three separate models will be used for each sub task: Random Forests, Support Vector Machines (SVMs), and an implementation of the VGG-16 Convolutional Neural Network (CNN). Both random forests and SVM models make use of a pretrained dlib 68 facial feature model to extract facial landmark coordinates as features. For task B, face shape and eye colour classification the aforementioned implementation of the VGG-16 CNN will be used.

Facial landmark data coordinates are a set of predefined points on a face that correspond to specific facial features such as eyes, nose and mouth. By using these facial landmark coordinates as features, models are able to focus on relevant information for gender and smile detection whilst reducing dimensionality and thus increase model performance. It is also notable that using facial features allows the models to be more robust to lighting and head pose. However, it must be noted that the dlib model may not be able to detect faces at all the angles present in the dataset. Specifically the dlib model was chosen is due to its low computational requirement as the model implemented in C++ and makes use of the Adam-Krylov optimization algorithm which is a gradient-free optimization algorithm meaning it does not compute the gradients or other derivatives and instead uses iterative methods to approximate the solution of a linear system of equations.

Random Forests is an ensemble learning method that combines multiple decision trees, where they use the concept of bagging (bootstrap aggregating), which is a technique which creates multiple subsets of the data by randomly sampling the data with replacement. Different subsets are used to train different decision trees and the results are combined for a final prediction. This method allows for random forests to improve upon decision trees by reducing variance which subsequently reduces overfitting. This reduction in overfitting is beneficial for the CelebA dataset due to it's relatively small size as models would overfit to reach high accuracy. Random Forests models have also been shown to be robust to variations in lighting conditions, head pose and high dimensional data which are all characteristics of the CelebA dataset. Random Forests also have the ability to determine feature importance, which is a measurement of how important each feature is on final predictions. This feature importance can be used to identify the most important features in the data and can be used to reduce the dimensionality of the CelebA dataset and Cartoon Set.

SVMs are a type of supervised learning algorithm which can be used for binary classification. e best hyperplane is the one that maximizes the margin, which refers to the distance between the hyperplane the closest data points from each class, which are known as support vectors. SVMs are able to handle non-linearly separable data by using kernel functions which map the original data to higher-dimensional space where the data becomes linearly separable. This is done using the kernel trick which maps the data to higher dimensional space where it becomes linearly separable. The most commonly used kernels are linear, polynomial, and radial basis function (RBF). In the context of the CelebA dataset, SVM's are able to handle high dimensional, no linearly separable data which is useful as neither the raw image or landmark features are linear. Through the kernel trick SVM's are able to make these features linear leading to produce a high performance model. It is also able to select important features which will be particularly useful for smile detection as the mouth landmarks will have a significantly larger influence than other landmarks.

CNN's are a type of deep learning algorithm which are well suited for image classification tasks. They consist of multiple layers of artificial neurons that are designed to process and analyze images by learning to recognize patterns and features in the data without the need of feature extraction this is beneficial for the CelebA dataset as the dlib facial feature extractor will b unable to extract features from all of the dataset due to this complexity, therefore leading to a reduction in performance which is unseen by the CNN model. As CNNs can handle high-dimensional feature spaces and complex data, making them well-suited for the CelebA dataset which contains images of real faces with various angles, lighting, and pose. As CNN's typically require large amounts of data for training they will be well suited to the Google Cartoon set to achieve a very high performance. As CNN's can also be fine-tuned on smaller datasets through

transfer learning where ... which is an option if the implemented VGG-16 model does not perform well. Unlike SVM's CNNs are able to be used for both binary and multiclass classification tasks making them suited for both tasks A and B.

4. IMPLEMENTATION

In this section I will provide a detailed implementation preprocessing and models: random forests, SVM and VGG-16 and discuss how they are adapted to each task.

General libraries use are: numpy for storing and manipulating data in numpy arrays, , matplotlib.pyplot to plot evaluation graphs such as learning rate, randomSearchCV for hyperparameter tuning is used throughout, shuffleSplit used for the learning curve function, sklearn accuracy metrics such as accuracy score, precision score, recall score, f1 score, learning curve and confusion matrix are used for evaluation, pandas for data manipulation and analysis.

Preprocessing

Preprocessing refers to the feature extraction of facial landmark coordinates to be used by the random forests and SVM models which are used for Tasks A1 and A2. The key libraries this section uses are: dlib, for detecting faces and predicting facial landmarks on images, opencv for confirming images from RBG to greyscale, os for pathing and keras.preprocessing.image to load and do image preprocessing in terms of smoothing.

The implementation contains 4 key functions: shape_to_np(which converts and stores the facial landmark coordinates, rect_to_bb() converts the dlib to the format (x, y, w, h) using openCV, run_dlib_shape() which loads the images, converts to greyscale and the neccessary datatypes, detects a face then landmarks, and returns the new image and landmard coordinates and finally the extract_features_labels(): which literates through every image and extracts and stores the facial landmark coordinates as well as extracting the relevant label into 2 numpy arrays. This code is repeated for both the training and test data.

It must be noted that only 4798 training images and 970 test images are used as facial landmark features extraction failed on 202 training and 30 test images, this will be same for both tasks A1 and A2 for random forests and SVM implementations.

This code is reused for random forests and svm in tasks A1 and A2 changing the parameters to extract the relevant labels. The code also works to reduce dimensionality as colour is not required for gender or smile detection.

Random Forests

Specific random forests module imports: Sklearn for the randomforestsclassifier algorithm.

Firstly, this program loads the numpy arrays which store facial landmark coordinate features and the relevant labels for both the training and test dataThe training data is not split into training and validation data as the RandomizedSearchCV function contains this functionality. The reason RandomizedSearchCV is used over GridSearchCV as the random search algorithm has been shown to be more computationally efficient especially when the selection range is large.

The hyperparameters to be tuned for the random forests classifier are n_estimators which refers to the number fo decision trees and max_depth which refers to depth of the forest, it must be noted as the number of trees increases the variance decreases and therefore decreases overfitting however increases computational costs. Increasing the max depth allows for the model to split the data into smaller subsets which can lead to the model learning the noise rather than generalising well, it also leads to an increase in computational cost. RandomizedSearchCV is used to test across n_trees from 50 to 300 in increments of 50 and max depth from 2 to 30 in increments of 2. The n_jobs parameter was used to set the number of CPU cores used at 4 to improve the speed of the results. This is done with the cross validation parameter equal to 10 therefore 500 images per training, this value was chosen as there is not very much data and dividing it further may lead to the model not generalising well. The optimal parameters were found to be max_depth = 20 and n_estimators = 100 with a best cross validated accuracy score of 87.05\% for gender detection and max_depth =16 and n_estimators = 200 with a best cross validated accuracy score of 88.91%. This model is then trained with these parameters on the entire training set and tested used the test set where the predictions are stored as y_pred. These predicted values are compared to the test labels and accuracy, precision, recall and f1_score are computed.

Finally, to evaluate the model a plot_learning_curve() function is created for better customization of the graphs. The learning curve function plots a graph of training sizes against mean training score and also plots mean testing scores on the same plot. The function also plots train sizes against mean fit times to evaluate the scalability of the model. A confusion matrix is also computed which will be discussed in the analysis.

Here is the random forest learning curve for gender detection and smile detection:

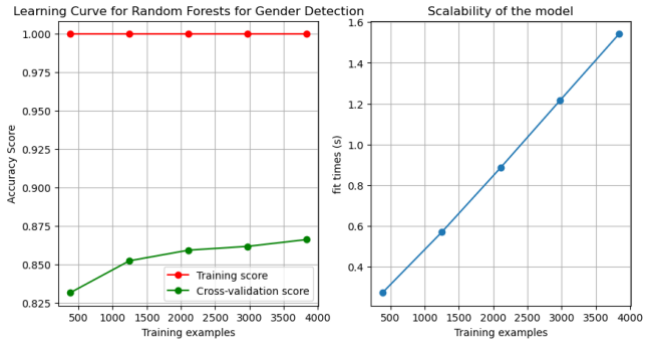

Figure 2 - Learning Curve for Random Forests on Gender Detection

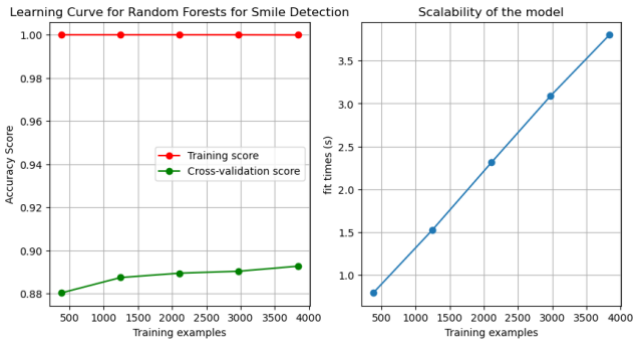

Figure 3 - Learning Curve for Random Forests on Smile Detection

Both these graphs show that the training model was overfitted as the training score and cross-validation score do not converge. However, these were the parameters received the best test accuracy scores regardless of overfitting. The scalability model also shows that fit times increase linearly with time, however as the fit times are so low this shouldn't be an issue. It can be seen that that as smile detection random forest model has a higher max depth and number of trees leading to the fit times per train size to be much longer.

Support Vector Machines

Specific SVM module imports: StandardScaler function from the sklearn.preprocessing library to scale both the training and test data to a mean of 0 and a variance of 1, this is done as SVMs are sensitive to the scale of the input data, therefore may lead to poor performance if not scaled. It also improves performance as the problem becomes well-conditioned to find a global minimum This scaling is applied to both training and test sets for consistency and **as to avoid data leakage as the model would learn the mean and variance of the training set and the test set will no longer be unseen.** SVC is also imported from the sklearn.svm library as that is the model used to test in the hyperparameter tuning, linearSVC is imported as LinearSVC is able to converge faster than SVC set to kernel linear as it uses liblinear compared to libSVM.

Firstly the code loads in the features and relevant labels using numpy arrays and scales them using standardScaler() for the reasons discussed above. Then large ranges for hyperparameter tuning is selected for linear, polynomial and radial bias function kernels as follows. C, the regularisation parameter is tested by generating 2- values between 10^-2 to 10^2 using the np.logspace function this range was selected for prevent overfitting by a very large regularisation parameter. This range of range for C was used for each kernel. The degree of the polynomial kernel was tested between 2 and 6 due to the high dimensionality of the data. The coef0 parameter was checked for 10 values between 10^-1 and 10^2 this range was chosen from initially picking a small range and measuring impact and adjusting from there. The gamma parameter was tested between 10^-3 to 10^3 with 6 evenly spaced values. Once these parameters are found they are put into the relevant models and tested for accuracy, precision, recall and f1_score are computed on the test set. The same learning curve and confusion matrix functions were used as the random forests module. This was repeated for both gender and smile labels yielding these learning curves. For the sake of conciseness only the best kernel in terms of accuracy of each task will be discussed, find other SVM kernel learning curves in supplementary materials.

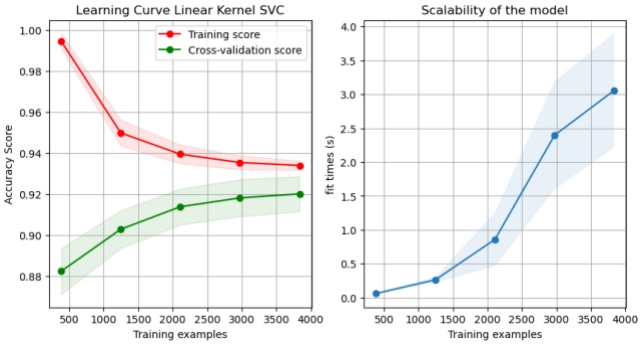

Figure 4 - Learning Curve for Linear SVM on Gender Detection

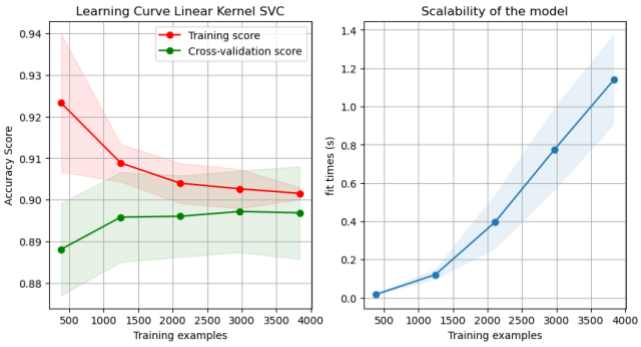

Figure 5 - Learning Curve for Linear SVM on Smile Detection

The code is provided on GitHub: https://github.com/yshv/ELEC0134.git

The specific CNN VGG16 imports: the keras.preprocessing.image library import for the ImageDataGenerator function to scale the images by 1/255 so the pixel values lie between 0 and 1 to help stabilize the deep learning model. Keras.models library is imported for the Sequential class to allow the creation of a sequential model which uses linear stack of layers where the output of a layer is the input the the next. The keras.layers library is imported for the following functions: Dense which is fully connected layer where each neuron is connected to the previous layer, Conv2D which is a 2D convolutional layer used for feature extraction, MaxPool2D which is used to down sample an input by using the max set of neurons in the previous layer, which is used to reduce computational complexity, reduce overfitting through parameter reduction and, allows each neuron to see larger parts of the input image. Flatten which is used to flatten the output to a 1D array which is used so the output of the Conv2D layers can be input into Dense layer. BatchNormalization is used to normalise the activations of the previous layer at each batch which helps stabilise the training process and improve overall performance. The tensorflow keras callbacks library is also imported for the earlystoping and model checkpoint functions which are used to incrementally save the model and stop training when convergence has been reached.

Firstly, a dataframe is created to store the labels and corresponding file name and labels are increased so the labels start at 1 rather than 0, this is convention. The labels are converted from type int64 to strings as this is expected from the imageDataGenerator class in keras. The dataframe is split into training and validation data, where the validation data is 20% of the training data. Two data generators are defined which scale the images by 1/255 so the pixel values lie between 0 and 1 and the the images are resized to 244 by 244 as is expected by the VGG16 arcitecture. The batch size of the data generators are set to 32 as a good balance between training times and memory requirements. Another data generator for the test data is also set up with the same parameters.

Keras callbacks Earlystoping and ModelCheckpoint are set up in the callbacks list and set to save the model and any subsequent models which improve in validation loss. Earlystopping is set up to stop after a certain number of no increases in the validation loss. This parameter is dependent on the application of the model as some tasks are much more difficult to converge such as the eye classification task which only stops after 10 epochs of no improvement in validation loss. A custom callback timehistory class is created to store the fit times after each epoch and amended to the callbacks list, this will be used to evaluate the scalability of the model.

The model architecture is composed of multiple convolutional, max pooling and dense layers. Initially the architecture starts with a convolutional layer with 16 filters and a kernel size of 3x3 with padding set to same. The initial filters is set to 16 so they model can start learning multiple features of the input image. The input shape is set to (224,244,3) which corresponds to the VGG16 input image size and the number of channels of the image which is RGB. The ReLU activation function is used to introduce non-linearity to the model to combat the vanishing gradient problem which occurs when parameters become very small for the model to learn as the ReLU function keeps the gradients large during backpropagation as well as enabling the model to learn more complex relationships between the input and output. Then BatchNormalization is used to normalise the inputs to the activation function for each batch to prevent internal covariate shift meaning as the inputs to a layer change it can also cause the gradients to change which may slow down to training process or make it unstable. The model applies a max pooling layer with size of 2x2 and stride of 2x2 which is used to down the output of the previous layer by taking a maximum value of a set of neurons in the previous layers, this is done to reduce spatial dimensions and parameters of the data which prevents overfitting. This architecture is stacked 4 more times with an increasing number of filters and max pooling layers. The filters increase in order: 16, 32, 64, 128 which allows the model to learn more complex features.

Finally, the output is passed to a flatten layer which flattens the output to a 1D array into a dense layer with 64 units and ReLU activation followed by a final dense layer with 5 units and a softmax activation function. The softmax activation function is sued to transform the output of the final dense layer into a probability distribution over 5 classes and the class is taken as the highest probability from the prediction mode. The Adam optimizer is used as it is able to handle large datasets and is computationally efficient, which uses first-order gradient based optimization of stochastic objective functions. The loss function used is cross-entropy loss for multiclass problems with the metric of accuracy. If this model is applied to task A it uses a final dense layer equal to 1 and activation of sigmoid which will output a probability between 0 and 1 and the loss function is binary_crossentropy. The models are evaluated using accuracy and test loss.
Gender detection learning curve:

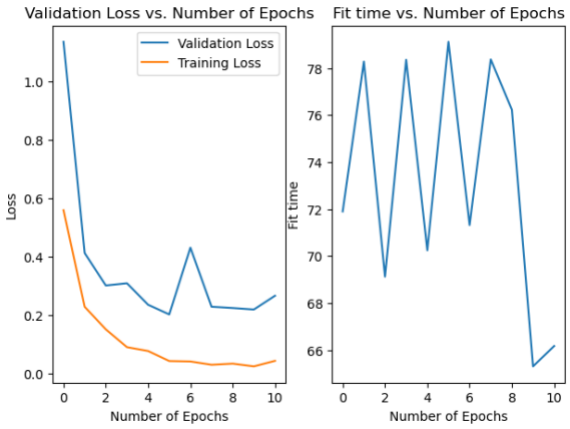

Figure 6 - Learning Curve for VGG16 CNN on Gender Detection

Smile detection learning curve:

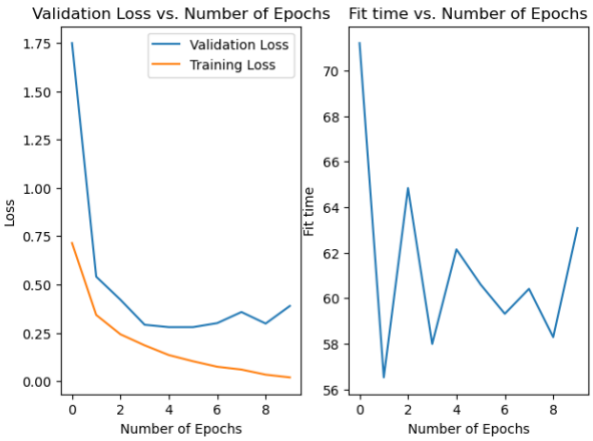

Figure 7- Learning Curve for VGG16 CNN on Smile Detection

It can be seen from both of these learning curves that the VGG16 model converged with a high variance. It is also notable that the model had a faster overall fit time in the smile detection compared to the gender detection.

Face shape classification learning curve:

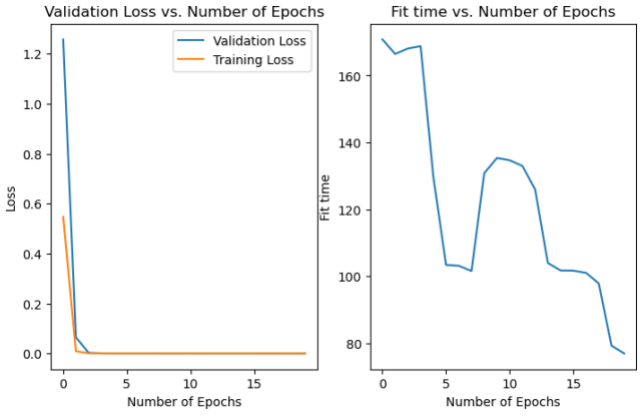

Figure 8 - Learning Curve for VGG16 CNN on Face Shape Classification

Eye colour classification learning curve:

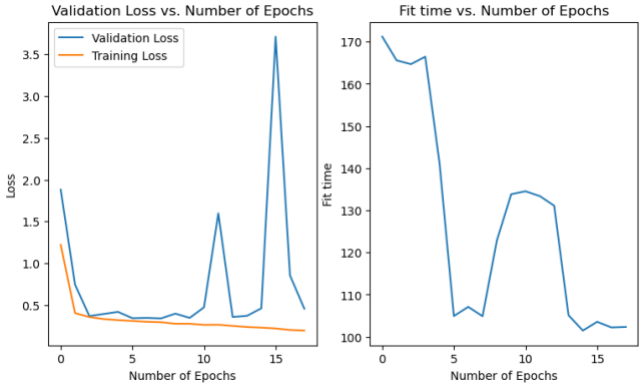

Figure 9 - Learning Curve for VGG16 CNN on Eye Colour Classification

With these graphs we can see all models converged within 0.2 loss of each other where the face shape had variance. We note the large fluctuations on the eye colour model indicating it was a sudden large change in the distribution of the dataset caused by random chance. We also note the fit time against number of epochs is gradually decreasing for each graph suggesting the models are efficient as time decreases upon increase in epochs. We also note the binary tasks converged in fewer epochs due to their simplicity.

5. EXPERIMENTAL RESULTS AND ANALYSIS

In this section I will present and analyse the results of my experiment on each task for facial image classification. We have employed 3 models: random forests, SVM and CNN and I have used the following evaluation metrics on the test set provided: accuracy, precision, recall and F1-score.

Table 1. A1 Gender Detection Results

	Accuracy	Precision	Recall	F1-Score
Random Forests	0.851	0.877	0.813	0.844
Linear SVM	0.925	**0.945**	0.900	0.922
Polynomial SVM	0.923	0.936	0.906	0.921
RBF SVM	0.923	0.938	0.904	0.921
VGG16 CNN	**0.938**	0.938	**0.938**	**0.938**

From Table 1, it can be seen that the SVM kernel models performed very similarly with accuracy scores ranging from 92.3% and 92.5%. The random forests model performed the worst with an accuracy of 85.1% this is likely due to the overfitting during the training phase. Unsurprisingly, the VGG16 CNN implementation performed the best model with the highest scores in all but precision. This is likely due to the increased amount of data taken in by the model as it used raw pixel data rather than only facial landmark coordinates. Overall, general performance on this task was high averaging accuracy over 90%.

The code is provided on GitHub: https://github.com/yshv/ELEC0134.git

Table 2. A2 Smile Detection Results

	Accuracy	Precision	Recall	F1-Score
Random Forests	0.897	0.911	0.885	0.898
Linear SVM	**0.904**	**0.914**	0.897	**0.906**
Polynomial SVM	0.901	0.903	**0.903**	0.903
RBF SVM	0.896	0.899	0.897	0.898
VGG16 CNN	0.879	0.885	0.879	0.879

From Table 2, it can be seen that the model performed slightly worse than gender detection. This is expected as it is harder to detect smiling especially on faces at an angle. This decrease in performance could also be due to the face all of the facial landmark data was fed in when it may not have been needed. The highest accuracy was achieved by the linear SVM model and in general the SVM models performed the best. This may be because the facial landmark data, specifically of the mouth may have been linearly separable. Surprisingly the CNN model performed the worst on this task with an accuracy of 87.9%. It is worth noting that the F1-score for all models is relatively low, which may indicate that the models are not well balanced in terms of precision and recall. This may indicate that the models are either too conservative or too liberal in its predictions.

Table 3. B1 Face Shape Classification Results

	Accuracy	Precision	Recall	F1-Score
Random Forests	0.865	0.837	0.902	0.868
VGG16 CNN	**1**	**1**	**1**	**1**

Table 3 shows that the VGG16 CNN model performed significantly better than the Random Forests model for the multiclass of face shape classification. The VGG16 CNN model achieved a perfect score of 1 for all evaluation metrics, including accuracy, precision, recall, and F1-score, indicating that it was able to classify the face shapes in the dataset with a certainty. On the other hand, the Random Forests model had an accuracy of 0.865, this was likely due to underfitting of the data and the images where facial features were not able to be extracted primarily due to glasses.

Table 4. B2 Eye Colour Classification Results

	Accuracy	Precision	Recall	F1-Score
VGG16 CNN	**0.849**	**0.852**	**0.849**	**0.849**

From the results in Table 4, it can be seen that the VGG16 is the only model which was used for the Eye colour classification It is to be noted that a significant portion of the data is useless as the colour of the eyes are obscured, therefore introducing a zero error. Random forests could not be implemented for eye colour classification as it used facial feature data which did not contain colour values.

The code is provided on GitHub: https://github.com/yshv/ELEC0134.git

6. CONCLUSION

In conclusion, this report has presented a thorough evaluation of various machine learning models for facial recognition tasks. The results showed that CNNs performed the best overall, with VGG16 achieving the highest scores on all tasks except for gender detection, where linear SVM achieved the highest accuracy. SVMs also performed well, with linear SVM achieving the highest scores on smile detection and polynomial and RBF SVMs performing similarly on gender detection. Random Forests, however, performed the worst overall with the lowest scores on all tasks. This suggests that CNNs are more suited for facial recognition tasks, and that linear SVMs can be used as an alternative when computational resources are limited.

For further work, it may be valuable to consider using unsupervised learning for the task of eye colour detection to remove the useless data in the Cartoon Set. It may also be also be useful to implement data augmentation, particularly on the more complex CelebA dataset or to compare the performance of pretrained models with our implementations.

7. REFERENCES

[1] S. Yang, C. Luo and C. a. X. T. Loy, "From Facial Parts Responses to Face Detection: A Deep Learning Approach," *IEEE International Conference on Computer Vision*, vol. ICCV, 2015.

[2] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," in *arXiv*, 2015.

[3] K. D and F. B, "Google Cartoon Set," Google, [Online]. Available: https://google.github.io/cartoonset/people.html.

[4] J. R. Quinlan, "Induction of Decision Trees," 1986, p. 81–106.

[5] L. Breiman, RANDOM FORESTS.

[6] E. Kremic and A. Subasi, "erformance of Random Forest and SVM in Face Recognition," *International Arab Journal of Information Technology* , vol. 13(2), 2015.

[7] C. M. Bishop, Pattern Recognition and Machine Learning, 2007.

[8] A. M. M Buda, "A systematic study of the class imbalance problem in convolutional neural networks," *Neural Networks*, vol. 106, pp. 249--259, 2018.

[9] V. N. Vapnik, The Nature of Statistical Learning Theory, Springer, 2000.

[10] C. L. C Hsu, "A comparison of methods for multiclass support vector machines," *EEE Transactions on Neural Networks*, vol. 13, pp. 415-425, 2002.

The code is provided on GitHub: https://github.com/yshv/ELEC0134.git