APPLIED MACHINE LEARNING SYSTEM ELEC0132 19/20 REPORT

SN: 19107985

uceenhe@myucl.ac.uk

ABSTRACT

This paper reports on the performance of four tasks, namely gender, emotion, face-shape and eye-colour classification problems. Each task was separately solved using a combination of face detection models, feature extraction methods, outlier detection methods and classification algorithms. The focus was on providing the highest possible performance accuracy, while taking the effect each step in the model has on the performance. Parameter optimisation techniques were also used to further increase the performance of the models. The following performances were achieved for each task respectively: 85%, 88.3%, 83.3% and 97.7%. The code to this project is made available *here*.

1. INTRODUCTION

In this paper, it will be shown that using a combination of face detection models, feature extraction methods, outlier detection methods and classification algorithms, facial image processing tasks can be solved. The following is brief description of the tasks and how they were solved.

Task A1 (gender classification) was solved as follows: A single-shot detector (SSD) based face detector was used to detect and extract the faces from images; Local binary pattern histograms were then extracted and used as input to the classifier; A support vector machine (SVM) using a radial basis function (RBF) kernel, along with parameter optimisation, was used to classify different genders.

Task A2 (emotion classification) was solved as follows: A SSD based face detector was used to detect and extract the faces from images; Facial landmarks, obtained from a facial landmark detector, were used to build the feature space; A SVM with a RBF kernel, along with parameter optimisation, was used to classify different emotions.

Task B1 (face shape classification) was solved as follows: Facial landmarks, obtained from a facial landmark detector, were used to build the feature space; Density Based Spatial Clustering of Applications with Noise (DBSCAN) was used to remove outliers from the data; A SVM using a RBF kernel, along with parameter optimisation, was used to classify different face shapes.

Task B2 (eye colour classification) was solved as follows: Facial landmarks, obtained from a facial landmark detector, were used to extract the eye from the image, from which an average RGB (Red,Green,Blue) was taken. This RGB value describes the entire feature space of the image; Inter-Quantile Range (IQR) outlier detection was used to remove outliers from the data; A SVM using a RBF kernel, along with parameter optimisation, was used to classify different face shapes.

This report is organised into 6 sections. Section 1, The Introduction, provides an overview of the methodologies used to solve the tasks. Section 2, The Literature Survey, provides incite into the current literature and the methodologies used to solve related problems/tasks. Section 3, The Description of Models, describes the models used for each task as well as the reasons for choosing them. Section 4, Implementation, provides the detailed implementation of the models used. Section 5, experimental Results and Analysis, describes and discusses the results obtained for each task. Section 6, Conclusion, provides a brief review of the results obtained as well as possible ways to improve the models. Section 7 and 8 are the References and Appendix respectively.

2. LITERATURE SURVEY

2.1. Current Works

2.1.1. Task A1: Gender Classification

Yang et al. [1] built an automatic face gender classification system, and did a comparative study on the effect that different texture normalisation[1] techniques and classifiers have on the performance of the system. They considered two kinds of affine mapping[2] and one Delaunay triangulation based warping[3] for texture normalisation. Furthermore, SVM-RBF, FLD (Fisher's Linear Discriminant) and Real Adaboost classifiers were compared. Their results showed that triangulation was best paired with Real Adaboost, achieving an error rate of 3.14%. Affine mapping was shown to be better suited for FLD and SVM achieving classification accuracies of 3.65% and 2.90% respectively.

Erno Mäkinen et al. [3] compared different state-of-the-art gender classification methods, along with a cascade face detector, to find out their actual reliability in solving gender classification problems. They compared a Neural Network (NN), SVM-RBF, mean and threshold Adaboost and two more modern variants, a SVM-RBF with local binary pattern (LBP) and Look-Up-Table (LUT) Adaboost. The top performing systems were as follows: the NN and LUT Adaboost both achieved an accuracy of 91.11%, mean Adaboost achieved an accuracy of 89.17%, and SVM-RBF with LBP achieved an accuracy of 86.28%.

Caifeng Shan [4] adapted the more classical SVM-RBF with LBP, in an attempt to achieve a more robust gender classification system. They adapted LBP by using Adaboost to select the best set of LBP features, forming a set of boosted LBP features. Together with SVM-RBF, they were able to achieve a classification accuracy of 94.81%. In more recent years, Betül Uzbaş et al. [5] also adapted the classical Polynomial-SVM with LBP to solve a gender based problem. They proposed a combination of features

[1] Texture normalisation is the process of manipulating the image pixels in such a way that each image in the data set are truly comparable. [1]

[2] Affine mapping is a linear mapping method that preserves points, straight lines, and planes. [2]

[3] Delaunay triangulation based warping is the process by which one warps each triangle texture of a given face to that of the mean shape. [1]

that have been extracted from the face, eye and lip regions by using a hybrid method of Local Binary Pattern and Gray-Level Co-occurrence Matrix. They were able to achieve a classification accuracy of 98%.

2.1.2. Task A2: Emotion Classification

Much like the work done by Shan [4] in 2012, Shan et al. [6] produced another piece of work in 2008, making use of the boosted LBP Histogram (LBPH) features, to classify facial expressions (emotions). They were able to achieve classification accuracies of 91.3% when pairing single-scale boosted LBPH features with an Adaboost classifier. Additionally, they were able to achieve a classification accuracy of 93.1% when pairing multi-scale boosted LBPH features with a Polynomial-SVM classifier.

Hua Gao et al. [7] developed a real-time non-intrusive monitoring system, which detects the emotional state of the driver by analysing facial expressions. They made use of two different feature types. The first being the Holistic Affine Warping (HAW) method, which normalises face images using the coordinates of the left eye and right eye. The second type uses the facial landmarks, as it preserves the geometrical information of the facial components and does not introduce additional artifacts. Each set of features was used along with a Linear-SVM. The system which used the facial landmarks outperformed the system which used the HAW features, and achieved a classification accuracy of 85%. It should be noted that to achieve these classification accuracies, pose normalisation was implemented.

Binh T. Nguyen et al. [8] developed an approach to detect human emotion in real time. Their experiments resulted in a classification accuracy of 70.65% when using a range of features from the facial landmarks, which were extracted from each image, paired with a SVM-RBF. They compared the results of the SVM-RBF with a Decision Tree classifier, Random Forest classifier and a Linear-SVM. The SVM-RBF system outperformed them all.

Luning. Li et al. [9] proposed an Active Appearance Model (AAM) based method to remove face shape noise for the purpose of improving facial expression recognition performance. The feature set, which is made up of facial geometry information, is extracted using AAM. Subsequently, based on the extracted features, an SVM classifier is used to classify the face shape. Next, a method using face show classification is implemented for facial expression classification, which improves the classification accuracy. They were able to achieve an average face shape classification accuracy of 89.4%.

2.1.3. Task B1: Face Shape Classification

Amir Zafar et al. [10] presented an automatic shape extraction and classification method for face and eye-ware shapes for the purpose of eye-ware recommendation systems. They identified key geometric feature sets, which were developed using facial landmarks, that can separate the face shape classes. Using the identified feature sets along with a nearest neighbour classifier, they were able to achieve classification accuracies of 80% and 99% for face and eye-ware shape respectively.

Kitsuchart Pasupa et al. [11] presented a novel system for a hairstyle recommendation system that is based on a face shape classifier. They used a combination of features, geometric features obtained from facial landmarks. The best set of features was then learned (through deep learning) using VGG-face dataset. They were able to achieve a classification accuracy of 70.33% using a SVM-RBF along with their learned features.

2.1.4. Task B2: Eye Colour Classification

While eye colour classification is well covered in current literature, due to the nature of the provided dataset (cartoon images), real life eye colour classification is unlikely to translate well to the task at hand. It is for this reason that literature covering the classification of colours is investigated instead.

Cheng-Jin Du et al. [12] developed an automatic pizza sauce spread classification system using colour vision and SVMs. After segmenting the image from the background, the image was converted from the Red-Green-Blue (RGB) colour space to the Hue-Saturation-Value (HSV) colour space. Using Principle Component Analysis (PCA), they reduced the HSV feature space. They were able to achieve a classification accuracy of 96.67% using the reduced feature space and a Polynomial-SVM.

Derek Magee et al. [13] presented an approach for efficient 'context' aware colour classification. The specific problem when they were looking to solve, is to classify the colour variations in microscope slides. They use a 'context vector', which provides context information, along with per-pixel RGB values as the features for their classifier. They used a Relevance Vector Machine (RVM) along with the feature set described to classify each colour.

Zezhi Chen et al. [14] presented an approach to segmenting moving road vehicles from the colour video data supplied by a stationary roadside CCTV cameras and classifying those vehicles in terms of vehicle type and colour. Since, the only step of interest is the colour classification step, only a review of that step will be presented. To perform the colour classification, the system used an 8-bin 3D colour histogram, where the RGB space is divided into equal size regions such that they are mapped to these regions into a histogram that can be populated by the colours of the pixels in the image, as the vector for Polynomial-SVM classification. They were able to achieve an average classification accuracy of 97.4%.

3. DESCRIPTION OF MODELS

3.1. Data Pre-processing

3.1.1. Task A1 and A2

The first step in solving tasks such as these ones, is the data pre-processing step, which consists of face detection and extraction. Based on the literature, deep learning face detection methods significantly outperform the more traditional face detectors. A popular deep learning method for face detection is SSD [15] and variations of SSD [16]. It was shown that SSD face detection models are capable of outperforming other stat-of-the-art deep learning face detection models. SSD is a simple method, compared to those which follow a similar strategy, because it encapsulates all computation into a single network. This in turn makes SSD easy to train and integrate into systems that require a detection component [15]. It is for these reasons that an SSD based face detector was used to help solve Tasks A1 and A2.

The face detector used is based on the work done by Lui et al. [15] in their paper 'SSD: Single Shot MultiBox Detector' and uses a Residual Neural Network (ResNet) at its backbone [17].

Lui et al. [15] presented an approach for detecting objects in images using a single deep neural network (DNN). Their approach consisted of a feed-forward convolutional network that produced a fixed-size collection of proposals and scores based on the presence of an object instance inside each of the bounding boxes. Subsequently, a filtering step is complete, which reduces the number of bounding boxes, to produce the final detections. The first network layers are of a standard architecture, which are used for high quality image classification. Additional accessory structure is then added to the network to produce detections.

After each face was detected, the region of interest (ROI) is cropped out from the original image and saved for later use. One might consider the need for facial alignment in a problem such as this one, however, after inspection, it was found that facial alignment had already occurred on the Celeb database images. Refer to figure 1 in 8, for proof of alignment.

3.2. Feature Extraction

3.2.1. Task A1

Local Binary Pattern (LBP) has become a popular approach for various image processing applications [6], due to its discriminative power and computational simplicity [18]. This was made evident from the literature described above, such as the works by Erno Mäkinen et al. [3], Caifeng Shan [4] and Betül Uzbaş et al. [5]. Each of these works used LBP to solve a form of gender classification. It is because of these reasons, that LBP is used as the feature extraction technique of choice for task A1.

LBP labels the pixels of an image by thresholding the neighborhood of each pixel and considers the result as a binary number [18], which can then be converted to its decimal equivalent. This process is depicted in figure 2 in section 8. R refers to the radius of the circle around the central pixel, and P refers to the number of sampling points on the circle. Using interpolation, any (P,R) combination is allowed, as it deals with pixels with co-ordinates that are non-integer values [18]. However, it is important to note that while increasing P will provide more information, the number of possible different labels increases (number of labels 2^P), and thus number of features increases, increasing the time take to process each image.

Once the LBP values are calculated, the frequency of each pattern is counted and a histogram created, which is subsequently used as the input feature for the image. This drastically reduces the feature space. However, in doing so, all spacial information is lost. To overcome this spacial information loss, an image may be broken up into several pieces/tiles, after which, the above described LBP process is performed on each tile of the image. Finally, the histograms, which each describe the local information of their respective images, are concatenated together to form a global description of the entire image [18].

3.2.2. Task A2 and B1

Facial landmarks are a set of points, usually located on the corners, tips or mid points of facial features [19], as seen in figure 6 in section 8. According to Yun Tie [19], one can categorise the landmark detection algorithms into two groups based on the type of features and the anthropomorphic information which is used: Geometric feature-based methods and Appearance-based methods.

The geometric feature-based methods utilise prior knowledge about the face position, and constrain the landmark search by heuristic rules that involve angles, distances, and areas. A number of these methods have had success in detecting facial features [19].The appearance-based methods use image filters to generate the facial features for the entire face or specific regions in the face image. The Active Shape Models (ASM) and Active Appearance Models (AAM) are two popular appearance-based methods.

Detection and tracking algorithms, which make use of facial landmarks, can be used for tasks such as facial expression recognition [19]. This adoption of facial landmarks in emotion classification problems can be seen in the literature [7] [8], as well as in face-shape classification problems [10] [11].

It is for these reasons that facial landmarks are used used to aid in soling Tasks A2 and B1. The used facial landmark detector and extractor from Dlib is an implementation of the work done by Vahid Kazemi et al. [20], in the paper 'One Millisecond Face Alignment with an Ensemble of Regression Trees'.

Kazemi et al. [20] were able to show that an ensemble of regression trees can be used to estimate the facial landmark positions directly from a subset of pixel intensities, achieving real-time performance with high quality predictions. They presented a framework based on gradient boosting for learning an ensemble of regression trees that optimises the loss and is able to handle data with missing labels. Additionally, they were able to show that by using proper priors they were able to exploit the structure of the image, which in turn helps with efficient feature selection [20].

3.2.3. Task B2

This is done using facial landmarks, where the eye is extracted from image based on the landmarks associated to the eye. The same landmark extraction technique utilised in Task A2 and B1 is used here.

Jeppe D.Andersen et al. [21] presented a new method for measuring the eye colour on a continuous scale. They used the custom designed software Digital Iris Analysis Tool (DIAT), which automatically extracts the iris from the image. Next, the software counted the number of blue (N_{bl}) and brown (N_{br}) pixels in the extracted iris image, and thus calculated the Pixel Index of the Eye (PIE-score), that is then used to describe the eyes colour. The PIE-score is calculated as follows:

$$PIE = \frac{N_{bl} - N_{br}}{N_{bl} + N_{br}} \tag{1}$$

The calculation of the PIE-score somewhat resembles the average colour of the iris. If the iris was all brown pixels, the PIE-score will be -1, if the iris was all blue pixels, the PIE-score will be 1 and if there are an equal number of blue and brown pixels detected, the PIE-score will be 0.

Andersen et al. [21] evaluated the PIE-scores by comparing them with human classifications of the same images. The results showed that the PIE-score correlated well with the human perception of eye colour.

Therefore, to solve Task B2, a technique inspired by the PIET-score to extract features from the image is used. Firstly, as described above, each image has an eye extracted and stored for later use. Once the eye extraction is complete, the average RGB value of each eye is calculated, and is used as the input features.

3.3. Outlier Detection and Removal

3.3.1. Task B1

Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise (DBSCAN) was proposed in 1996 by Martin Ester et al. [22], as means to meet what he considered the requirements for clustering algorithms: little knowledge of the domain is required to determine input parameters, the ability to discover clusters of arbitrary shape and good efficiency on large databases. At the time, no well-known clustering technique was able to provide these requirements. DBSCAN was presented as a density based clustering technique, which is designed to discover clusters of arbitrary shape. Additionally, DBSCAN is able to find a cluster which is completely surrounded by different clusters and is robust towards outlier detection [23] as well as was found to be significantly more efficient than the other well-known clustering techniques. It is for these reasons that DBSCAN is now one of the most commonly used clustering techniques used [23], subsequently, the reason DBSCAN is used to help solve Task B1.

Firstly, its important to define the rules which govern how the DBSCAN clustering technique works [22].

Definition 1: The Eps-neighbourhood of a point, $N_{Eps}(p)$, is defined by $N_{Eps}(p) = q \in D|dist(p,q) \leq Eps$, where Eps is the radius of the neighbourhood and D be a database of points.

Definition 2: A point p is directly density-reachable from a point q , if $p \in N_{Eps}(q)$ and $|N_{Eps}(q)| \geq MinPts$, where MinPts is the minimum number of points which have to be present in the Eps-neighbourhood to be considered a cluster.

Definition 3: A point p is density-reachable from point q if there exists a chains of points $p_1, ..., p_n$, where $p_1 = p$ and $p_n = q$ such that p_{i+1} is directly density-reachable from p_i.

Definition 4: A point p is density-connected to a point q, if there is a point m such that both p and q are density-reachable from m.

Definition 5: Cluster C is a non-empty subset of D satisfying the following conditions:

1. $\vee$ p,q: If p $\in$ C and q is density-reachable from p, then q $\in$ C.

2. $\vee$ p,q $\in$ C: p is density-connected to q.

Definition 6: Let $C_1, ..., C_k$ be the clusters of the database D with regards to parameters Eps_i and $MinPts_i$ where $i = 1, ..., k$, such that points not belonging to any cluster C_i, is considered an outlier.

Using the above definitions, the following two steps are taken to build a cluster:

1. Choose an arbitrary point p from the database D. If point p meets the condition to be a core point, continue onto the next step. Otherwise, repeat step 1.

2. Retrieve all points that are density-reachable and density connected to the core point p.

3.3.2. Task B2

To make an informed decision on which outlier detection technique should be employed for Task B2, the distribution of each eye colour is plotted on an RGB set of axis. Additionally, bounding boxes are used to clearly identify the outliers of each eye colour. This can be seen in figure 7 in section 8. Each distribution has a cluster of outliers near the origin of the distribution. Additionally, there are several other outliers, which can be seen in the distribution, where their RGB values are slightly shifted from the main cluster.

After inspecting the dataset, it became clear what was the cause of these outliers, namely, characters wearing sunglasses or tinted glasses. Exmaples of these can be seen in figure 8 in section 8.

Based on the above, the chosen outlier detection technique for Task B2 is the IQR method. IQR is a simple to implement, statistic-based method, which fits the structure of the data well, since it is clear from figure 7 that the majority outliers will fall outside the IQR threshold and subsequently be labeled as outliers. Refer to figure 9 in section 8 for a crude example.

3.4. Classification

Support Vector Machines (SVMs) are one of the most efficient machine learning algorithms [24]. They have been used for a wide range of tasks, such as image, face and speech recognition, face detection, text categorisation and fault detection [24]. This wide adoption can be seen in the literature review, section 2, where current works, Shan [4], Nguyen et al. [8], Pasupa et al. [11] and Cheng-Jin Du et al. [12] are seen using SVMs to solve a range of problems, such as gender, emotion, face-shape and colour classification. It is because of these reasons that the SVM has been used to solve all four tasks: A1, A2, B1, B2.

SVMs can be used to solve standard binary classification, multi-class classification and regression problems [25]. SVM uses a discriminant hyper-plane to separate classes [26]. The selected hyper-plane is the one that maximises the margins (maximisation problem). The margin is defined as the distance from the nearest training points (also known as the support vectors) to the hyper-plane.

The use of support vectors, instead of all points in the dataset, in calculating the optimal hyper-plane greatly reduces the required computation as only a small fraction of the data is required to compute the optimal solution [25]. It is important to note that in most problems, the classes are likely to be not separable (unlike the simplified example shown in figure 10 above), and thus the maximisation problem is unsolvable. To overcome this problem, a new variable, the the regularisation parameter (C) [26] is introduced, one which relaxes the strictness of the boundary condition of the hyper-plane. The larger C is, the more strict the boundary is, and thus there are fewer mis-classifications. If C is too large, there is risk that the system will become overfitted to the training data, and in contrast, if C is too small, the system risks being under-fitted and not separating the classes well enough [25][26].

Traditionally, SVM used a linear decision boundary to separate the classes. However, this greatly reduces the number of problems which a traditional SVM can solve [25]. This problem was overcome by introducing what is referred to as the 'kernel trick' [26], which uses non-linear kernels to 'map' the data into

a higher dimension. It is important to not that the mapping occurs implicitly, meaning that the number of observations does not change, and thus the same number of observations in a higher dimensional space is more sparse, making the data more likely to be linearly separable [25].

The kernel function is often referred to as K(x, y), where the most popular choice for a non-linear kernel is the RBF kernel [25], which has the following function:

$$K(x, y) = e^{\frac{(-||x-y||)}{2\sigma^2}}) \quad (2)$$

where σ is the width of the Gaussian bump.

The RBF kernels' popularity can also be seen in the literature review, where most SVM models used, made use of the RBF kernel.

There is no equivalent multi-class SVM algorithm which one can use to solve classification problems with more that two classes [25]. To over come this, two different approaches have been proposed: One-Versus-All (OVA) and One-Versus-One (OVO).

OVA organises the classes in such a way that multiple two-class SVM models are trained using one class against the rest of the classes (which are given the same classification label). This is done for every class in the dataset. Therefore, if there are K number of classes, there are K number of binary classifiers [26].

OVO organises the classes in such a way that multiple two class SVM models are trained using a single class against another single class (the rest of the classes are ignored). Therefore, if there are K number of classes, there a total of K(K - 1)/2 binary classifiers [26].

In both cases, a sort of voting scheme is deployed to determine the which class and unlabeled point belongs to.

4. IMPLEMENTATION

For Tasks A1 and A2, a dataset (celeba dataset) which contains 5000 labeled facial images (178x218 pixels) of celebrities. Each image is described by two labels, gender and emotion.

For Tasks B1 and B2, a dataset (cartoon_set dataset) which contains 10000 labeled cartoon facial images (500x500 pixels). Each image is described by two labels, face shape and eye colour.

The following libraries, as well as some other auxiliary libraries, were used to implement the models described in section 3.

1. NumPy: A library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

2. Pandas: A library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series.

3. Scikit-learn: Also known as sklear, is a machine learning library for the Python programming language. It contains modules capable of various classification, regression and clustering algorithms and is designed to work in conjunction with the Python numerical and scientific libraries NumPy and SciPy.

4. Scikit-image: Also known as skimage, is a collection of algorithms for image processing.

5. OpenCV: A library of programming functions mainly aimed at real-time computer vision.

6. Dlib: A general purpose cross-platform library.

4.1. Task A1: Gender Classification

4.1.1. Data pre-processing

OpenCV's new Deep Neural Network model (dnn) is used to implement the pre-processing method described in section 3.1. In particular, dnn's functions *readNetFromCaffe()* and *blobFromImage()* were used. Their implementation can be seen in figures 11 and 12 in section 8.

In figure 11, *readNetFromCaffe()* is used to read an already made neural network which is capable of face detection. It uses a *.prototxt* file, which defines the model architecture, such as the layers themselves and a *.caffemodel* file, which contains the associated weight for the layers [27].

In figure 12, *blobFromImage()* is used to do mean subtraction, scaling and optionally channel swapping of the image. The output is then used as the input to the net (*net.setInput()*), and subsequently face detection is run (*net.forward()*) and the detections returned.

The network returns a list of 'faces' which have been detected in the image, along with a corresponding confidence and (x,y) coordinates of the associated bounding box. Since the images all have a single face of interest, the detection with the highest confidence is used as the detected face for that image. The result of the pre-processing step is as follows; 2496 cropped images of female celebrities and 2489 cropped images of male celebrities i.e. 99.7% of all faces were detected (0.3% loss in samples).

4.1.2. Feature extraction

Functions *local_binary_pattern()* and *histogram()*, from Scikit-images and NumPy respectively, were used to implement the feature extraction described in section 3.2.

Each extracted face is broken up into tiles of size 25x25 pixels, such that an LBP histogram can be extracted from each tile. The LBP functions is defined by the number of sample points taken along the specified radius and the method type. After the Uniform LBP patterns are extracted from each tile, the histogram function counts the frequency of each pattern. The histogram is then normalised and appended to the 'global' histogram, which is then subsequently used as the input to the classification algorithm. Refer to figure 13 in section 8 for code snippet.

Each image is now described by 1296 features, thus substantially reducing the feature space from 178x218 pixels. This data set it then split into training and validation data, 75% and 25% respectively i.e. the training input data had the shape(3738, 1296) and the validation input data had the shape(1247, 1296).

4.1.3. Classification

Using Scikit-learns' svm module, in particular svm's *SVC()* function, the classification model described in section 3.4 was implemented.

As discussed, an SVM model using a RBF kernel is implemented. As a result of this, parameters such as C, Gamma and the decision function shape are required to further describe the SVM model. Therefore, to determine the best values and shape, a grid-search of a specified parameter-space incorporating cross-validation (CV), is used. Skicit-learn has a module named model-selection, which has a function called *GridSearchCV()*.

GridSearchCV() takes in as its paramters the classification model, the parameter-space and the number of CVs. Once the grid-search object has been defined, the training data is 'fitted', and all the possible combinations within the parameter-space is tested. Refer to figure 14 in section 8 for code snippet. After the 'fitting' is complete, the best parameter combination can be determined. The following 'best' parameter combination was achieved:

$$C : 10.0; decision\_function\_shape : OVO; gamma : 0.1$$

Once the best parameter combination has been determined, a Learning Curve is plotted such that the bias and variance of the model can be determined. This is done using model-selections' function *learning_curve()*, which takes parameters that describe the classifier, the training data and labels, the CV technique, and the size of the training samples for each iteration of CV. Refer to figure 16 in section 8 for code snippet. The training set is used as the input data to *learning_curve()*.

Consider figure 17 in section 8, where the learning curve generated using the 'best' parameters obtained from the grid search can be seen. It is clear from the learning curve that the model using the 'best' parameters has very low bias (almost zero), but very high variance. This means that the current model has overfitted to the training data. To overcome this, the regularisation parameter C is reduced, thus making the model more general. Based on the learning curves for C = 5 and c = 1 in figure 17, C = 1 is chosen as the new C value used to describe the SVM model. This is final model chosen is:

$$C : 1.0; decision\_function\_shape : OVO; gamma : 0.1$$

The final SVM model is then trained using the training data, and tested using the validation data.

4.2. Task A2: Emotion Classification

4.2.1. Data pre-processing

The same data pre-processing implementation was used for Task A2 as in Task A1.

4.2.2. Feature extraction

The result of the pre-processing step in Task A2 is as follow; 2498 cropped images of smiling celebrities and 2487 cropped images of not-smiling celebrities i.e. 99.7% of all faces were detected (0.3% loss in samples). This data is then used in the feature extraction process.

As mentioned in section 3.2, the facial landmarks predictor used is from the Dlib library. The predictor is defined using the *shape_predictor()* function, with a .dat file as input. The .dat file describes the land mark structure used in the predictor. It must be noted that the *shape_predictor* object requires both an image and an object of type *dlib.rectangle*. The rectangle object is the bounding box of the detected face. For simplicity, Dlibs *get_frontal_face_detector()* is used as it returns the correct object type. Refer to figure 18 in section 8 for code snippet.

Now, using the facial landmarks, a list of features are developed based on them, as per [28]. The list of features includes the follow:

1. All the (x,y) co-ordinates of the facial landmarks.
2. The distance from each facial landmark and the 'centre' of the image.

The new set of data is now made of images described by 268 features, drastically reducing the feature space from 178x218 pixels. The set is then split into training and validation data, 75% and 25% respectively i.e. the training input data had the shape(2614, 268) and the validation input data had the shape(872, 268). There was roughly a 29% loss in input images during this step (29% of the inputted images were detected as having no faces, due to the Dlib face detector) and thus an overall loss of about 31%.

4.2.3. Classification

As described for Task A1, the *SVC()* function is used to implement the SVM classification model, while *GridSearch()()* is used to to find the 'best' combination of C, Gamma and the decision function shape. The following 'best' parameter combination was achieved:

$$C : 1000; decision\_function\_shape : OVO; gamma : scale$$

Note that the gamma value 'scale' is defined as:

$$Gamma = \frac{1}{(n_{features})(Var(X))}$$

Learning curves are subsequently plotted for C = 1000, 500, 100 , 50 as seen in figure 19 in section 8. While the learning curve achieved using the 'best' combination of parameters shows both a relatively low bias and variance, the C parameter is very large, and is likely to result in overfitting of the training data. Thus smaller C values were tested. It is seen that in decreasing the C parameter, the bias increases and the variance decreases, with a minimal drop in training and validation accuracy. It was found that by testing models using $C = 100, 50$ on the validations dataset, the model with $C = 100$ performed the best, and thus the final model chosen is:

$$C : 100; decision\_function\_shape : OVO; gamma : scale$$

4.3. Task B1: Face Shape Classification

4.3.1. Feature extraction

As in Task A2, the facial landmarks predictor used is from the Dlib library. The only variation is how the landmark points are used to develop a feature list. Refer to figure 24 in section 8 for code snippet.

The list of features developed are as follows (features 2-5 are based on the work done by [29].):

1. All (x,y) co-ordinates of the facial landmarks that correspond to the jawline.

2. Forehead: distance from the peak of one eyebrow to the peak of the other one.

3. Cheekbones: across the upper cheeks; starting and ending at the outer corner of each eye.

4. Jawline: measure of the jaw across your face at its widest point.

5. Face length: distance from the center of the hairline to the tip of the chin.

The FACIAL_LANDMARKS_IDX provides a simple way of extracting the required landmarks from the prediction as each item maps to a specific set of landmarks points and their index's (item 7 of the FACIAL_LANDMARKS_IDX corresponds to the index of the landmarks associated to the Jaw).

The new set of data has the shape (8090,38), where each image was now described by 38 features. This is a massive decrease in feature size from 500x500 pixels. There was roughly a 20% loss in data (20% of the inputted images were detected as having no faces). Shapes (classes) 0, 1, 2, 3, 4 had the respective losses 21%, 22%, 15%, 18% and 20%.

4.3.2. Outlier detection

Using Skicit-learns' cluster library, specifically, the *DBSCAN()* function, the DBSCN method described in section 3.3 is implemented. *DBSCAN()* is described by eps and MinPts. Refer to figure 22 in section 8 for code snippet.

To determine the ideal eps value, the mean distance between every point and its nearest 50 neighbours is calculated. The ordered means are then plotted per class. The eps value is chosen as the point of inflection, as this represents the mean distance between the 'closest points' in the class. As seen in figure 21 in section 8, the mean distance at which inflection start is around 5 for all classes, thus this is the chosen eps value.

According to Jörg Sander et al. [30], MinPts can be chosen to be $(2 \times n_{features})$. Based off this, outliers were detected for a range of $MinPts = i \times n_{features}, i = 1, 2, 3, 4$. It was found that $i = 3$ performed the best. Using this combination of parameters, the outliers per class (for both training and validation data were found and removed.

The resulting size of the dataset after outlier removal is 7548 images. This is a total of 542 images removed due to outlier detection (7%), and now a total 25% of the original data has been removed. The remaining data is then split into training and validation data, 75% and 25% respectively i.e. the training input data had the shape(5661, 38) and the validation input data had the shape(1887, 38).

4.3.3. Classification

Using *SVC()* and *GridSearch()()* t he following 'best' parameter combination was achieved for Task B1:

$$C : 1000, decision\_function\_shape : OVO, gamma : 0.001$$

Learning curves are subsequently plotted for C = 1000, 500, 100 ,50 as seen in figure 23. While the learning curve achieved using the 'best' combination of parameters shows both a relatively low bias and variance, as in Task A2, the C parameter is still very large, and is likely to result in overfitting of the training data. Thus smaller C values were tested. It is seen that in decreasing the C parameter, the bias increases and the variance decreases, with an increasing drop in training and validation accuracy. Based on the learning curves, $C = 100$ is chosen as it is the best balance between variance, bias and accuracy. Thus the following are the new selected parameters for the model:

$$C : 100; decision\_function\_shape : OVO; gamma : 0.001$$

The final SVM model is then trained using the training data, and tested using the validation data.

4.4. Task B2: Eye Colour Classification

4.4.1. Data pre-processing

The facial landmarks predictor used is from the Dlib library to extract the facial landmarks. After which, as described in section 3.1, the landmarks are used to extract the eye (ROI) from the image. In specific, item 4 of the FACIAL_LANDMARKS_IDX, which corresponds to the right eye is use. Each eye is then saved for later use. There were a total of 8090 eyes saved (19% loss).

4.4.2. Feature extraction

The process of feature extraction described in section 3.2 is implemented by taking the mean RGB value of each image, as seen in figure 25. The feature is soley made up of R-,G-, and B-Channels, thus creating a data set of the shape (8090,3). The feature space has been tremendously reduces from 500x500 pixels.

4.4.3. Outlier detection

To implement the IQR method described in section 3.3, NumPy's *quantile()* function is used, as seen in figure 26in section 8. The inputs *quantile()* is the array and the quantile fraction. Since quantile 1 and 3 are of interest, fractions 0.25 and 0.75 are used respectively. The IQ range is then determined and used to set the upper and lower thresholds. A point is then considered an outlier if it does not fall within these bounds. This process is completed for each R-,G-, and B-channel for every eye colour class.

The resulting size of the dataset after outlier removal is 6920 images. This is a total of 1170 images removed due to outlier detection (14%), and now a total 31% of the original data has been removed. the remaining data is the split into training and validation data, 75% and 25% respectively i.e. the training input data had the shape(5190, 3) and the validation input data had the shape(1730, 3).

4.4.4. Classification

Using *SVC()* and *GridSearch()()* the following 'best' parameter combination was achieved for Task B2:

$$C : 100, decision\_function\_shape : OVO, gamma : 0.001$$

Learning curves are subsequently plotted for C = 100, 50, 10 as seen in figure 27 in section 8. The learning curve achieved using the 'best' combination of parameters shows both a fairly low bias and variance, while the C parameter is of a respectable size. For comparison sake smaller C values were tested. It is seen

that in decreasing the C parameter, there is minor increase in bias and minor decrease in variance, with a very small drop in training and validation accuracy. As C gets smaller though, the Learning Curves starts to warp for small training sizes with a major drop in accuracy. Therefore, the 'best' combination parameters is kept for the final model. The final model is then trained using the training data, and tested using the validation data.

5. EXPERIMENTAL RESULTS AND ANALYSIS

5.1. Results

The following results were obtained using the described final models in section 4.

Task	Pre-proc.	Feat. Extraction	Out. Det.	Classifier
A1	Face detec.	LBPH	N/A	SVM_{RBF}
A2	Face detec.	Facial landmarks	N/A	SVM_{RBF}
B1	N/A	Facial landmarks	DBSCAN	SVM_{RBF}
B2	ROI extr.	Average RGB	IQR	SVM_{RBF}

Table 1. Table showing complete models used for each task.

Task	Val Acc	Test Acc	Lit. Acc
A1	0.866	0.850	0.863-0.980
A2	0.877	0.883	0.707-0.931
B1	0.793	0.833	0.703-0.894
B2	0.984	0.977	0.967-0.974

Table 2. Table showing classification accuracies achieved on validation and test data, as well the range of accuracies from reviewed literature.

5.2. Analysis

The OpenCV DNN face detector, on average, missed 0.3% of the faces it was shown. The Dlib face detector on the other hand, managed to missed an average of 23% of the faces it was shown. This means that Task A2, B1 and B2 all suffered a large loss in data, thus potentially altering their performance. While the feature space achieved in Task A1, 1296 features, using LBPH is substantially smaller than if the raw pixels were used as features (178x218), it is still quite large and could result in overfitting, had the training set been smaller (curse of dimentionality).

Outlier detection is useful when it is removing noise and unrelated data from the dataset. However, in Tasks B1 and B2, the outliers were predominantly due to either the cartoon characters wearing glasses (B2) or the due to the characters have facial hair (B1), which could be considered important information.

The overall performance of each model in solving each task, as seen in 5.1, are all of a satisfactory level when compared to the results achieved in current literature 5.1. Additionally, when comparing the models performance on the validation and the test data, it can be seen that the models are well defined such that no overfitting has taken place.

6. CONCLUSION

It this paper, models were developed and implemented to solve four tasks; Gender Classification (A1), Emotion Classification (A2), Face Shape Classification (B1) and Eye Colour Classification (B2). It was found that classification accuracies of 85%, 88.3%, 83.3% and 97.7% respectively, were achievable when using the implemented models. To further improve the models and their performance it would be beneficial to try using a more reliable face detector (such as the OpenCV DNN face detector) when using facial landmark detection (such as Dlibs facial landmark detector) as this would greatly reduce the number of missed face defections and is likely to thus increase the models performance. Additionally, the feature space of some of the models were large, and the performance could likely have been improved had a feature space reduction method been implemented, like PCA. Lastly, it would be useful to, instead of disregarding the outliers detected in Tasks B1 and B2, label them as anomalies, such that the model can learn to detect them and thus inform the user accordingly.

7. REFERENCES

[1] Zhiguang Yang, Ming Li, and Haizhou Ai, "An experimental study on automatic face gender classification," in *18th International Conference on Pattern Recognition (ICPR'06)*. IEEE, 2006, vol. 3, pp. 1099–1102.

[2] MathWorks, "Affine transformation," .

[3] Erno Mäkinen and Roope Raisamo, "An experimental comparison of gender classification methods," *pattern recognition letters*, vol. 29, no. 10, pp. 1544–1556, 2008.

[4] Caifeng Shan, "Learning local binary patterns for gender classification on real-world face images," *Pattern Recognition Letters*, vol. 33, no. 4, pp. 431–437, 2012.

[5] Betül Uzbaş and Ahmet Arslan, "Gender classification from face images by using local binary pattern and gray-level cooccurrence matrix," in *Tenth International Conference on Machine Vision (ICMV 2017)*. International Society for Optics and Photonics, 2018, vol. 10696, p. 106960Z.

[6] Caifeng Shan and Tommaso Gritti, "Learning discriminative lbp-histogram bins for facial expression recognition.," .

[7] H. Gao, A. Yüce, and J. Thiran, "Detecting emotional stress from facial expressions for driving safety," in *2014 IEEE International Conference on Image Processing (ICIP)*, Oct 2014, pp. 5961–5965.

[8] Binh Nguyen, Minh Trinh, Tan Phan, and Hien Nguyen, "An efficient real-time emotion detection using camera and facial landmarks," 04 2017, pp. 251–255.

[9] Luning. Li, Jaehyun So, Hyun-Chool Shin, and Youngjoon Han, "An aam-based face shape classification method used for facial expression recognition," .

[10] Amir Zafar and T. Popa, "Face and eye-ware classification using geometric features for a data-driven eye-ware recommendation system," 2016.

[11] Kitsuchart Pasupa, Wisuwat Sunhem, and Chu Kiong Loo, "A hybrid approach to building face shape classifier for hairstyle recommender system," *Expert Systems with Applications*, vol. 120, pp. 14–32, 2019.

[12] Cheng-Jin Du and Da-Wen Sun, "Pizza sauce spread classification using colour vision and support vector machines," *Journal of Food Engineering*, vol. 66, no. 2, pp. 137–145, 2005.

[13] Derek R Magee, Darren Treanor, Phattthanaphong Chomphuwiset, and Philip Quirke, "Context aware colour classification in digital microscopy.," in *MIUA*, 2011, pp. 135–140.

[14] Zezhi Chen, Nick Pears, Michael Freeman, and Jim Austin, "A gaussian mixture model and support vector machine approach to vehicle type and colour classification," *IET Intelligent Transport Systems*, vol. 8, no. 2, pp. 135–144, 2013.

[15] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37.

[16] Yuqian Zhou, Ding Liu, and Thomas Huang, "Survey of face detection on low-quality images," in *2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018)*. IEEE, 2018, pp. 769–773.

[17] Vikas Gupta, "Home," Oct 2018.

[18] M. Pietikäinen, "Local Binary Patterns," *Scholarpedia*, vol. 5, no. 3, pp. 9775, 2010, revision #188481.

[19] Yun Tie and Ling Guan, "Automatic landmark point detection and tracking for human facial expressions," *EURASIP Journal on Image and Video Processing*, vol. 2013, no. 1, pp. 8, 2013.

[20] V. Kazemi and J. Sullivan, "One millisecond face alignment with an ensemble of regression trees," 2014, pp. 1867–1874, IEEE Computer Society.

[21] Jeppe D Andersen, Peter Johansen, Stine Harder, Susanne R Christoffersen, Mikaela C Delgado, Sarah T Henriksen, Mette M Nielsen, Erik Sørensen, Henrik Ullum, Thomas Hansen, et al., "Genetic analyses of the human eye colours using a novel objective method for eye colour classification," *Forensic Science International: Genetics*, vol. 7, no. 5, pp. 508–515, 2013.

[22] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al., "A density-based algorithm for discovering clusters in large spatial databases with noise.," in *Kdd*, 1996, vol. 96, pp. 226–231.

[23] Shilpa Dang, "Performance evaluation of clustering algorithm using different datasets," *IJARCSMS*, vol. 3, pp. 167–173, 01 2015.

[24] Sasan Karamizadeh, Shahidan M Abdullah, Mehran Halimi, Jafar Shayan, and Mohammad Javad Rajabi, "Advantage and drawback of support vector machine functionality," in *2014 International Conference on Computer, Communications, and Control Technology (I4CT)*. 2014, pp. 63–65, IEEE.

[25] Nick Guenther and Matthias Schonlau, "Support vector machines," *The Stata Journal*, vol. 16, no. 4, pp. 917–937, 2016.

[26] Nathan Herr, "Investigation into sEMG Signal Processing for Gesture Classification," pp. 83–92, 2018.

[27] Adrian Rosebrock, "Face detection with opencv and deep learning," Nov 2019.

[28] Palkab, "Emotion recognition using facial landmarks, python, dlib and opencv," .

[29] Akik Kothekar, "Face shape classifier," Lenskart.

[30] Jörg Sander, Martin Ester, Hans-Peter Kriegel, and Xiaowei Xu, "Density-based clustering in spatial databases: The algorithm gdbscan and its applications," *Data mining and knowledge discovery*, vol. 2, no. 2, pp. 169–194, 1998.

[31] Adrian Rosebrock, "Facial landmarks with dlib, opencv, and python," Jun 2019.

[32] Saptashwa Bhattacharyya, "Dbscan algorithm: Complete guide and application with python scikit-learn," Dec 2019.

8. APPENDIX

Fig. 1. Figures from Celeb dataset showing effect of facial alignment. The drag marks are due to the alignment algorithm 'filling' in the space once the images position is is aligned.

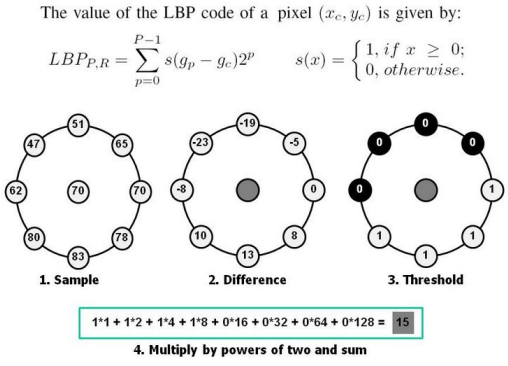

Fig. 2. Figure showing the LBP process steps, where the neighbouring points are 1st sampled, then the difference between their intensities is taken and finally the thresholding, where the LBP pattern is formed. Taken from [18].

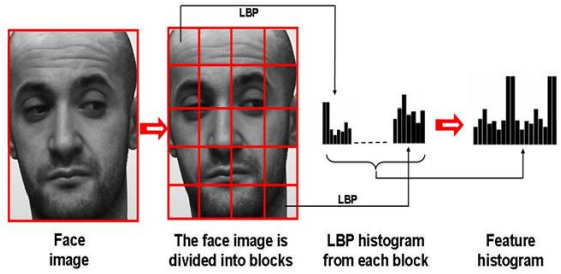

Fig. 3. Figure showing the LBP region separation process, where the image is broken up into different tiles, and an LBP histogram calculated for each tile. The histograms are then concatenated to form the final feature histogram. Taken from [18].

Fig. 4. Figure showing all 68 facial landmarks returned by Dlibs landmark detector using 'shape_predictor_68_face_landmarks.dat'. Taken from [31].

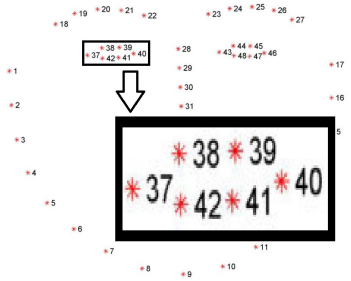

Fig. 5. Figure showing bounding box for eye extraction, where points 37-42 are used to defined it. Adapted from [31]

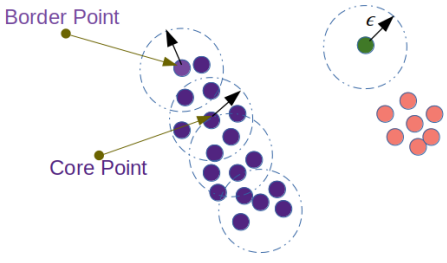

Fig. 6. Figure showing a cluster, which is made up of border and core points, and outliers . A core point is one whose Eps-neighbourhood is greater than or equal to MinPts. A border point is one who does not have a Eps-neighbourhood greater than or equal to MinPts, but is density-reachable to a core point. Taken from [32].

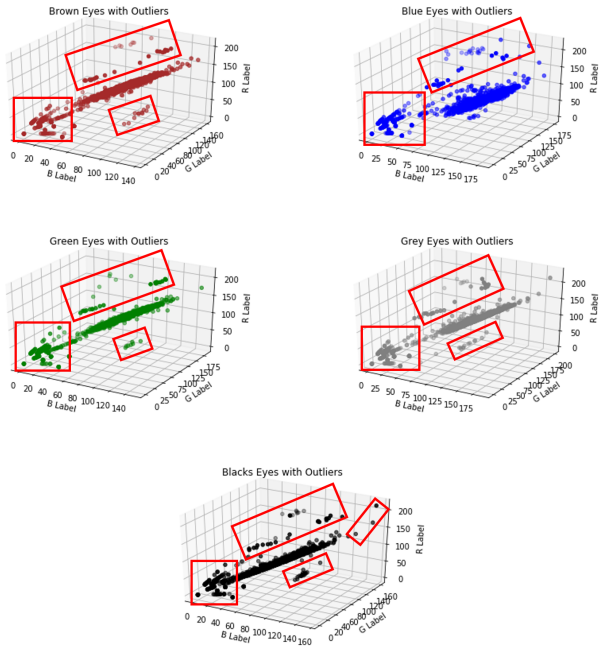

Fig. 7. Figures showing data distribution of each eye colour. Bounding boxes, in red, are used to identify outliers.

Fig. 8. Figures showing characters with blue eyes wearing sunglasses or tinted glasses, with 'mock' eye extraction to convince the reader of their effect on feature values.

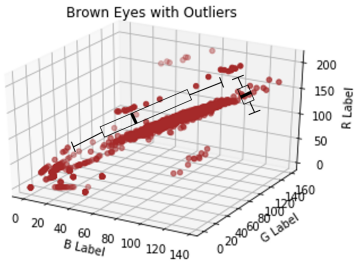

Fig. 9. Figure showing intuition for selecting IQR for outlier detection, where 'mock' box and whisker plots are drawn around the main cluster.

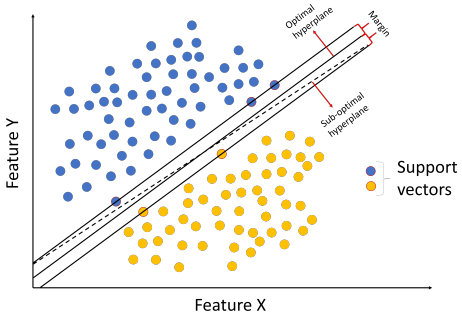

Fig. 10. Figure showing the hyper-plane and support vectors (shown with red borders) used in the SVM method. Note that the dashed line is a sub-optimal hyper-plane. Taken from [26]

```
net = cv2.dnn.readNetFromCaffe("../deep-learning-face-detection/deploy.prototxt.txt",
                               "../deep-learning-face-detection/res10_300x300_ssd_iter_140000.caffemodel")
```

Fig. 11. Figure showing code snippet using *readNetFromCaffe()* to create face detector network.

```
BGR_ave = image.mean(axis=(0,1))
blob = cv2.dnn.blobFromImage(cv2.resize(image, (300, 300)), 1.0, (300, 300), (BGR_ave[2], BGR_ave[1], BGR_ave[0]))
net.setInput(blob)
detections = net.forward()
```

Fig. 12. Figure showing code snippet using *blobFromImage()* .and the created network to detect faces from images

```
tiles = [image[x:x+M,y:y+N] for x in range(0,image.shape[0],M) for y in range(0,image.shape[1],N)]
radius = 3
no_points = 8*radius  # Number of points to be considered as neighbourers
eps=1e-7
global_hist_norm = []
for tile in tiles:
    lbp = local_binary_pattern(tile, no_points, radius, method='uniform') # Uniform LBP is used
    (hist, _) = np.histogram(lbp.ravel(), bins=np.arange(0, no_points + 2), range=(0, no_points + 2))
    hist = hist.astype("float")
    hist /= (hist.sum() + eps)
    hist_norm = hist
    global_hist_norm.append(hist_norm)
s = np.asarray(global_hist_norm).shape
global_hist_norm = np.reshape(global_hist_norm,(s[0]*s[1]))
```

Fig. 13. Figure showing code snippet using *local_binary_pattern()* and *histogram()* to create required LBPH feature space for Task A1.

```
clf = SVC(kernel='rbf',C = c , gamma = gamma, tol = tol)
training_data, prediction_data, training_labels, prediction_labels = train_test_split(image_inputs,image_labels)
clf.fit(training_data, training_labels)
yPredict = clf.predict(prediction_data)
accuracy_score(prediction_labels, yPredict)
```

Fig. 14. Figure showing code snippet using Scikit-learns SVC function used to train and subsequently predict the labels of the validation set.

```
tuned_parameters = [{'kernel': ['rbf'], 'gamma': [0.001, 0.01, 0.1, 1,10,100],
                     'C': [0.001, 0.01, 0.1, 1, 10,100,1000],
                     'decision_function_shape':['OVO','OVA']}]

clf = GridSearchCV(SVC(), tuned_parameters, cv=5, scoring='%s_macro' % score)
clf.fit(X_train, y_train)
clf.best_params_
```

Fig. 15. Figure showing code snippet of SVM parameter optimisation using *GridSearchCV()*.

```
train_sizes, train_scores, test_scores = learning_curve(estimator, X, y, cv=cv, n_jobs=n_jobs,
                    train_sizes=train_sizes)
```

Fig. 16. Figure showing code snippet of *learning_curve()* implementation.

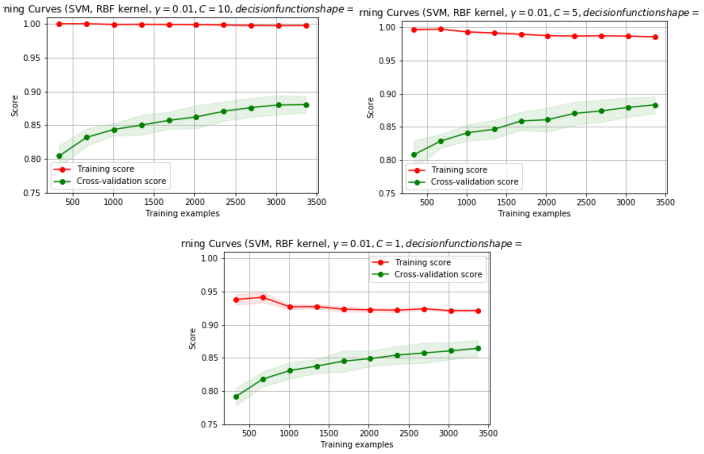

Fig. 17. Figures showing learning curves for C = 10, 5, 1 for Task A1.

```
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor("shape_predictor_68_face_landmarks.dat") #Or set this to whatever you named the downloaded file
detections = detector(image, 1)
shape = predictor(image, d) #Draw Facial Landmarks with the predictor class
```

Fig. 18. Figure showing code snippet of Dlibs face detector and shape predictor used to extract facial landmarks from the image.

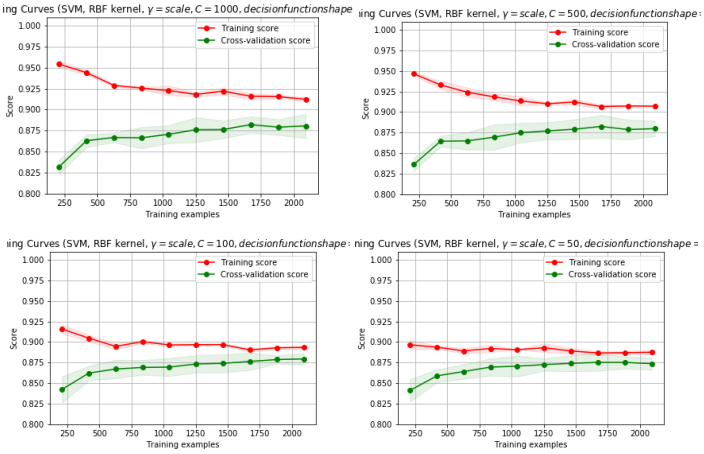

Fig. 19. Figures showing learning curves for C = 1000, 500, 100, 50 for Task A2.

```
name, (i,j) = items[7] #the 7th item in the FACIAL_LANDMARKS_IDXS is the jaw line
landmarks_vectorised = shape[i:j].tolist()
forehead = euclid_distance(shape[19], shape[24])
cheekbones = euclid_distance(shape[36], shape[45])
jawline = euclid_distance(shape[0], shape[16])
midpoint_eyebrows = midpoint(shape[19], shape[24])
face_lenght = euclid_distance(midpoint_eyebrows, shape[8])
```

Fig. 20. Figure showing code snippet of features extraction methods used for Task B1.

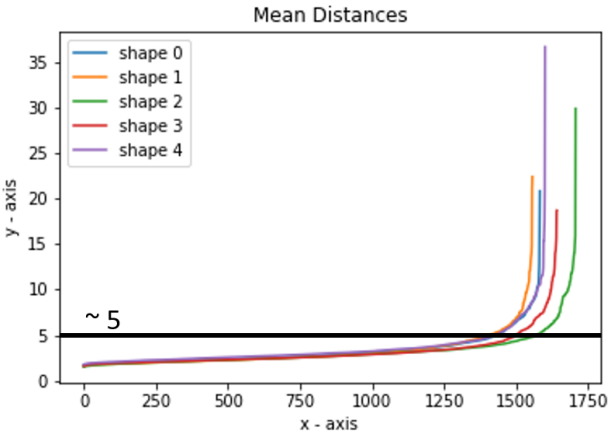

Fig. 21. Figure showing mean distances (between each point and its 50 nearest neighbours) plotted in ascending order for all classes. Solid black line shows point of inflexion.

```
shape0_dbscan = DBSCAN(eps = 5, min_samples = 38*3).fit(shape0_inputs)
shape0_outlier_labels = shape0_dbscan.labels_

shape1_dbscan = DBSCAN(eps = 5, min_samples = 38*3).fit(shape1_inputs)
shape1_outlier_labels = shape1_dbscan.labels_

shape2_dbscan = DBSCAN(eps = 5, min_samples = 38*3).fit(shape2_inputs)
shape2_outlier_labels = shape2_dbscan.labels_

shape3_dbscan = DBSCAN(eps = 5, min_samples = 38*3).fit(shape3_inputs)
shape3_outlier_labels = shape3_dbscan.labels_

shape4_dbscan = DBSCAN(eps = 5, min_samples = 38*3).fit(shape4_inputs)
shape4_outlier_labels = shape4_dbscan.labels_
```

Fig. 22. Figure showing code snippet of labeling the outliers for Task B1 using *DBSCAN()*. These points will subsequently be removed.

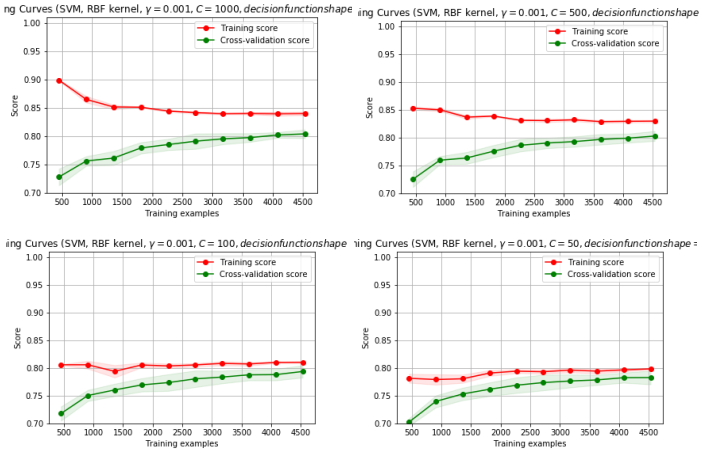

Fig. 23. Figures showing learning curves for C = 1000, 500, 100, 50 for Task B1.

```
items = list(face_utils.FACIAL_LANDMARKS_IDXS.items())
name, (i,j) = items[4] #the 4th item in the FACIAL_LANDMARKS_IDXS is the right eye
(x, y, w, h) = cv2.boundingRect(np.array([shape[i:j]]))
roi = image[y:y + h, x:x + w]
```

Fig. 24. Figure showing code snippet of extraction of the right eye for Task B2 where the 4th item from the FACIAL_LANDMARKS_IDXS is used.

```
image.mean(axis=(0,1))
```

Fig. 25. Figure showing code snippet of calculating the mean RGB value of an image of Task B2.

```
brown_Q1_b = np.quantile(df_brown_inputs['b'].values,0.25)
brown_Q3_b = np.quantile(df_brown_inputs['b'].values,0.75)
brown_IQR_b = brown_Q3_b - brown_Q1_b
brown_Minimum_b = brown_Q1_b - 1.5*brown_IQR_b
brown_Maximum_b = brown_Q3_b + 1.5*brown_IQR_b

brown_Q1_g = np.quantile(df_brown_inputs['g'].values,0.25)
brown_Q3_g = np.quantile(df_brown_inputs['g'].values,0.75)
brown_IQR_g = brown_Q3_g - brown_Q1_g
brown_Minimum_g = brown_Q1_g - 1.5*brown_IQR_g
brown_Maximum_g = brown_Q3_g + 1.5*brown_IQR_g

brown_Q1_r = np.quantile(df_brown_inputs['r'].values,0.25)
brown_Q3_r = np.quantile(df_brown_inputs['r'].values,0.75)
brown_IQR_r = brown_Q3_r - brown_Q1_r
brown_Minimum_r = brown_Q1_r - 1.5*brown_IQR_r
brown_Maximum_r = brown_Q3_r + 1.5*brown_IQR_r


for i, row in enumerate(brown_inputs):
    keep = (((row[0] > brown_Minimum_b) and (row[0] < brown_Maximum_b))
            and ((row[1] > brown_Minimum_g) and (row[1] < brown_Maximum_g))
            and ((row[2] > brown_Minimum_r) and (row[2] < brown_Maximum_r)))
    if(keep):
        idx_keep_brown.append(i)
```

Fig. 26. Figure showing code snippet of IQR implementation for a single class for Task B2. Both 1st and 3rd quantiles are calculated and thus the inter quantile range determined. Subsequently, using the IQR range, a thresholding technique is used to remove oultliers.

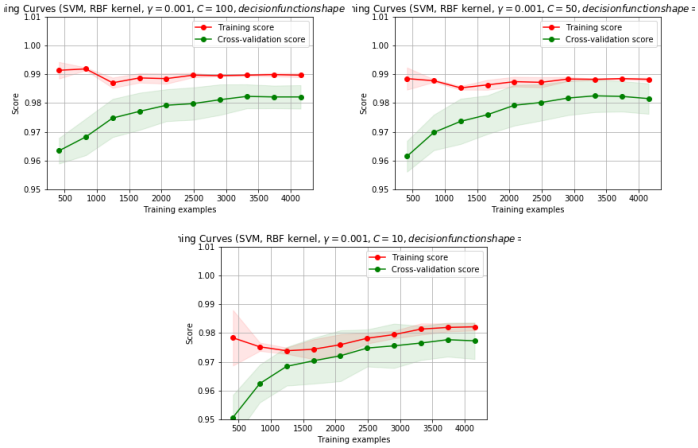

Fig. 27. Figures showing learning curves for C = 100, 50, 10 for Task B2.