I CAN SHOW YOU THE WORLD: TEACHING COMPUTERS HOW TO CLASSIFY IMAGES

APPLIED MACHINE LEARNING SYSTEM ELEC0132 19/20 REPORT

SN: 18154195

ABSTRACT

In an attempt to understand the underlying concepts behind machine learning systems, here we presented with four tasks in the area of computer vision. The first two tasks are binary classification problems, covering gender classification (A1) and smile detection (A2). The next tasks are multiclass classification which covers the issues of face shape (B1) and eye colour (B2) classification. Two distinct algorithms are used to detect the region of interest (ROI), namely facial landmark detector and Haar-like features. The same goes with feature extraction, where various signals are utilised across different tasks, such as the raw pixels, coordinate features, and histogram of the local binary pattern (LBP). These different features become the input of an SVM classifier with linear kernel. During the inference process, each task gained different performance, ranging from 67% for B1 to 98% for task B2.

Index Terms— image recognition, SVM, LBP, ROI detection, facial landmark detector

1. INTRODUCTION

In this report, we discuss the step by step of how to tackle four supervised machine learning tasks within the domain of computer vision. The first two problems are gender detection (task A1) and smile detection (task A2) based on the dataset of face images of celebrities [1]. Two other problems belong to multiclass classification domain: the face shape classification (task B1) and hair colour detection (task B2) based on human cartoon figures dataset [2].

The training process consists of two main pipelines: feature extraction and the actual classification process that takes the training data to tune the model. The feature extraction pipeline is a series of steps which include detecting region of interest (ROI) within an image, then transform it into more compact features without losing the essential signals related to the task goal.

The source code of the project is available as a GitHub project at https://github.com/donyeun/AMLSassignment19_20 and a Dropbox shared folder

As the features are fed into the classifier, the grid search and cross-validation method are used for the objective of finding the best model that fits the data. After the inference process, which runs the resulted model with the testing set, verified performance measurement is then reported in the form of accuracy score.

This report is organised into several sections. In the following Section 2, we explore briefly the fundamental theories of the algorithm and previous works related to the tasks. Section 3 discusses the description of models that justifies the choice of using SVM as the classifier. The algorithm implementation is highlighted in Section 4. Section 5 reported some intriguing findings and analysis, and the report is ended with Section 6 as the conclusion.

2. LITERATURE SURVEY

In this section, we highlight some of the background theory behind the algorithms we picked. The essential algorithms are also discussed, such as the Haar-like features and facial landmark detector to discover the ROI. Local binary pattern, support vector machine and some common machine learning preprocessing are also described in the following subsections.

2.1. Haar-like Features

Haar-like feature-based cascade classifier was proposed by Viola and Jones [3]. This classifier is often used for object detection, such as human face detection [4] and traffic light detection [5]. Not only in the domain of image, but the Haar-like features can also be implemented within the context of real-time video data [6].

Haar-like features were trained from a large quantity of positive and negative images of particular objects that matter. This rectangle features are usually in the form of a box, where it has multiple smaller boxes inside that can either be coloured black or white. The value of these rectangle features is calculated by summing the region inside the white boxes, subtracted by the sum of pixel values within black boxes. These entire values will then be passed into an adaptive boosting classifier [3].

2.2. Facial Landmark Detector

The facial landmark detector is another object detection algorithm within computer vision domain. The detector is the work of Kazemi et al. [7], that takes a sparse subset of pixel intensities. The next step is that the detector discovers the facial landmark positions within the image. The model was based on regression tree ensembles as the classifier [7]. This implementation of the face detector seems to perform better than Haar-like features at detecting frontal human faces [8].

The Dlib Python library's facial landmark detector [9] is the actual implementation of this algorithm in Python programming language. This implementation detects 68 coordinate points of facial feature edges, that marked the region of face, mouth, eyes and jawline within a human facial image.

2.3. Local Binary Pattern

Local binary pattern (LBP) has been widely used in various computer vision problems [10]. This feature extraction algorithm was being proposed for rotation invariant texture operator for an image in greyscale [11]. Firstly the algorithm will label either 1 or 0 for each neighbouring p pixel within the window. It compares the value of each pixel within a window to the centre value, as stated within equation 1 and 2, where g_c is the grey value of p neighbours.

$$LBP_{P,R} = \sum_{p=0}^{p-1} s(g_p - g_c)2^p \tag{1}$$

$$s(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases} \tag{2}$$

The $s(x)$ is the threshold function. It yields $s(x) = 1$ if the point is greater than or equal to the centre point of the window; otherwise, it returned $s(x) = 0$. The next step is to translate the eight neighbours of 1s and 0s into a decimal number. This single number will then become an LBP value of that particular region.

LBP can be used as a single feature, such as in the work of Lian et al. [12] that used LBP to extract facial features in multi-view gender classification task. Furthermore, LBP can also be accompanied by other feature extraction methods, such as within the work of Alexandre et al. [13] where LBP was being used along with other features such as intensity and edge direction. This work resulted in 91.19% reported accuracy on gender classification task by making use of SVM with the linear kernel as classifier.

2.4. Feature Scaling

For numerical stability, we will need to centre and scale the numerical value of the dataset. This preprocessing will resulted in dataset having a standard normal distribution with mean equals to zero and standard deviation to one. The formula is as depicted in equation 3, where z_i is the new value, x_i is the original numerical value, with μ and σ being the mean and standard deviation of the dataset, respectively. In the implementation in Python programming language, Scikit-learn provides $StandardScaler()$ function for this matter [14].

$$z_i = \frac{x_i - \mu}{\sigma} \tag{3}$$

2.5. Support Vector Machine

Support Vector Machine (SVM) is fundamentally a binary classification algorithm. Each point in space is a training datum. It will then use hyperplane as large margin separator that divides two classes of data [15].

SVM is a common classifier that is used on image classification task such as gender classification [10]. SVM with non-linear kernels are often used, although some researches have also been conducted by using simple SVM with linear kernel and still resulted in the remarkable result.

For example, research conducted by Ramon et al. [16] using simple LBP features and SVM-linear, gained 97.64% accuracy on MORPH dataset that consists of nearly 18,000 images of frontal human faces. This research also investigated two other distinct datasets with the same training pipelines in order to understand the performance stability regardless of the dataset. In the end, they concluded that their classification pipelines performance was dependant to the dataset being used. Although they did not gain the same accuracy score, yet they managed to get 90.60% on LFW dataset and 82.65% on The Image of Groups dataset.

Another example of previous works on image classification using SVM with linear kernel is [17] where 83% accuracy score was reported in gender classification with LBP histogram features. Another research conducted by Alexandre [13] reported 99.07% accuracy for its SVM linear classifier to detect gender on FERET dataset. The work of Kazemi et al. also exhibits the good performance of SVM linear classifier in detecting smile from an image [8].

3. DESCRIPTION OF MODELS

In this section, we describe the two main components of each task: the classifier and the feature extraction.

3.1. Linear SVM as Classifier

Across the four tasks, we use SVM with a linear kernel as the classifier. The use of merely one kernel on classifying using SVM might not be ideal, but as mentioned in Section 2.5, similar previous works on different datasets have been done by using SVM linear with reasonable accuracy scores [10] [13] [16] [17].

Linear SVM was chosen mainly because of two reasons: not only it gave a reasonable performance, this kernel also needed less computing power relative to other kernels. Conducting experiments with several k-fold cross-validations, with several selections of kernels, is computationally expensive, let alone the kernel itself has several hyperparameters such as C and γ for the non-linear kernel.

While some efforts have been made to gain more computing power during the execution of this project, such as utilising faculty's and university's cloud and SSH [18], and making use of Google Colaboratory [19], but these methods ended up did not work as expected. This was mainly due to practical constrains, such as run time limitation, as the model tuning requires significant execution time.

In multiclass classification, the $LinearSVC()$ function uses one-vs-all method as opposed to one-vs-one. This resulted in LinearSVC has fewer models to fit (only N models) as opposed to $SVC()$ that needs to fits $\frac{N*(N-1)}{2}$ models, where N is the number of class within a dataset [14]. SVM with linear kernel has one hyperparameter, that is the C, or soft margin constant [20]. In each task, a linear SVM with L2 norm as a penalty is used as the classifier.

We do not use a more sophisticated and popular deep learning-based approaches for this project, mainly to practice the sense of what is happening behind the process and to get a better understanding of crafting the features, as opposed to see the whole machine learning process (feature selection, hyperparameter tuning, the fitting process) merely as *black box* [21]. Because of this, the critical pipeline of this project is the feature extraction process. In the following sections, we will discuss how to formulate the features at each task.

3.2. Feature Extraction

In general, there are two main challenges concerning feature engineering within this project: how to detect the ROI, and how to perform the feature engineering.

ROI detection is important because the objective of each task is not interested in the whole image, but merely in a specific area of the image. In task A2: smiling detection, for example, we are only interested in the region of the lip. This way, it makes more sense if we feed the lip sub-image to the classifier, as opposed to the entire image by itself.

The next process then is to process the cropped output image from the ROI detector. An image is actually a 2-dimensional array consist of pixels. Each pixel consists of RGB number, which is an array with three values: the red, green, and blue colour intensity. However, in task A1: gender classification, for example, the colour intensity of every single pixel within the image is not particularly useful to determine the gender label. This is one of the reasons on why further feature engineering process is needed.

Within several next sections, we will discuss the interesting choice of ROI detector and features for each task.

3.2.1. Task A1 (Gender Classification) and A2 (Smile Detection)

The general flow of feature extraction is shown in Fig. 1. In the first two tasks, the feature extraction process starts with ROI detection. In task A1, the ROI is the square frame face area of a person, while in task A2, the ROI is the frame of the lip or mouth region.

Then after successfully recognised the region that matters, we crop the image, convert it into greyscale, then do the LBP calculation. As mentioned within Section 2.3, LBP is widely used in an image classification task, and some previous research on gender classification listed within survey paper by Ng. et al. [10] also used this feature and gained reasonable performance. The parameter that works best on our datasets are LBP with $p = 8$ and $r = 1$, which confirms the findings of [16].

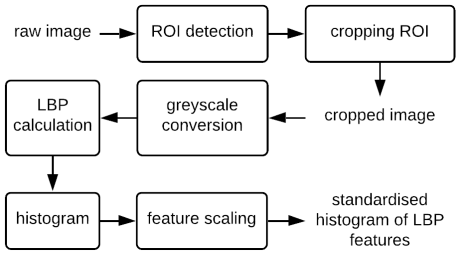

Fig. 1. Feature extraction for task A1 and task A2

3.2.2. Task B1 (Face Shape Multi-class Classification)

The flow of the feature extraction process within task B1 is shown in Fig. 2. What makes it different from the prior two tasks is that B1 uses the (x,y) coordinates of jawline as the region of interest. As we get the features that we want, further preprocessing such as feature scaling using z-score normalisation is executed.

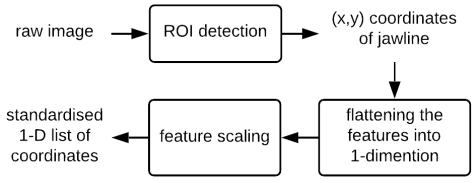

Fig. 2. Feature extraction for task B1

3.2.3. Task B2 (Detecting Five Types of Eye Colour)

Fig. 3 shows the feature extraction process for task B2. Crafting the feature for this task is the easiest amongst the four, as the objective to classify the eye colour is attached to the raw image as an array of RGB pixels. The feature of this task is the flattened 1-D array of the ROI sub-image's colour intensity.

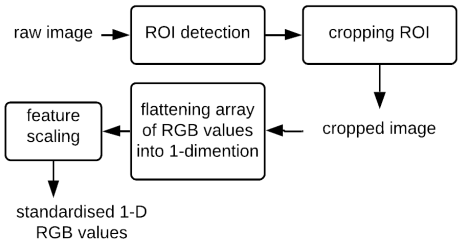

Fig. 3. Feature extraction for task B2

4. IMPLEMENTATION

In this section, we describe the detailed implementation of the tasks, starting from the description of datasets, followed by a brief explanation regarding the training pipeline, and several remarks on each task.

4.1. Datasets Description

The first dataset is the *celeba* [1]. It contains 5,000 images of celebrities that are mainly front-facing with good lighting yet have low resolution. All of the images within the dataset have the same resolution, which is 178 x 218 pixels. Each image has $gender$ and $smiling$ attributes that are being used as the label for binary classification task A1 (gender classification) and task A2 (smile detection), respectively.

The second dataset is 10,000 images from *cartoonset* [2], containing randomly generated avatar. The $face\_shape$ attribute acted as the label for task B1 (face shape classification), and eye_color as the label for task B2 (eye colour classification). Both of these are multiclass classification problems. Each image in the dataset is precisely 500 x 500 pixels in size.

Fig. 4, which is bar charts built using Matplotlib [22], shows the nearly equal frequency of labels for each dataset, indicating no further process needed to even out the distribution.

4.2. Training Pipeline

As we used SVM with a linear kernel as the classifier, the only hyperparameter to tune was the C, or soft margin constant [20].

The dataset was divided into train, validation, and test set as depicted in Fig. 5. Firstly, the entire data was chopped into either training or testing set, with a proportion of 80% and 20% respectively. This testing set was kept and only used during the inference process. During the training process, the training data was divided into two parts: the actual training set and validation set. These two sets were being used during the training process to tune the model and to seek the best hyperparameter.

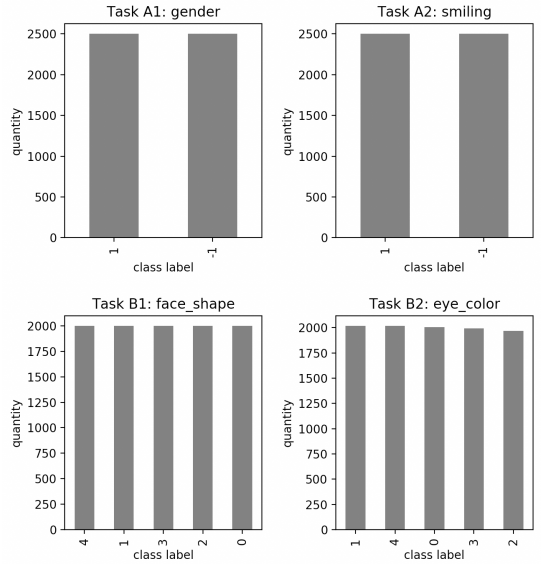

Fig. 4. The histogram for each task indicating an equal amount of data for each label

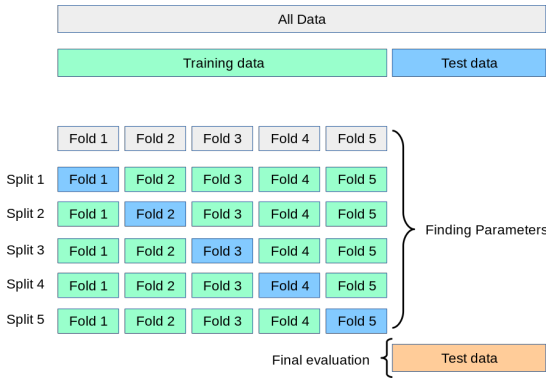

Fig. 5. Image from Scikit-learn [23], depicted the representation of cross-validation process during hyperparameter tuning

In implementation, the Scikit-learn's $train\_test\_split()$ and $GridSearchCV()$ were being used to execute the hyperparameter tuning pipeline [14]. The pseudocode of the training process is shown on Listing 1. Another contributing Python libraries on the training process, including Scikit-image [24] and OpenCV [25] as computer vision libraries and NumPy [26] for numerical computing in Python.

```
pipeline = make_pipeline(
    StandardScaler(),
    LinearSVC())

param_candidates = {
    'linearsvc__C' : c_params_list}

clf = GridSearchCV(
    pipeline,
    param_candidates,
```

```
    cv = k_cv)

clf.fit(X, Y)
```

Listing 1. Main training pipeline code

4.3. Task A1 (Gender Classification) Remarks

Task	ROI detector	Dataset	Total detected	%
A1	Dlib's 68-point	Celeba	4,831 / 5,000	97%
A2	Dlib's 68-point	Celeba	4,831 / 5,000	97%
B1	Dlib's 68-point	Cartoonset	8,393 / 10,000	84%
B2	Haar cascade	Cartoonset	7,133 / 10,000	71%

Table 1. ROI detection algorithm and its success rate of detecting the ROI within the image. It is safe to note, however, that the percentage does not indicate the quality of the detection itself.

In general, facial landmark detector [9] acted as ROI detector for all the tasks, except for task B2 that used Haar-like features [3]. Not only it has been proven to be better at detecting face than the Haar-like features [8], but it was also apparent that after several attempts, facial landmark detector worked best in the tasks' datasets on ROI detection.

The detection rate of each ROI detector algorithm for each task is presented in Table 1. For task A1 and A2, since both of them used the same image dataset and the same ROI detection algorithm, the detection rate resulted in the same value at 97%. However, one thing to note is that these detection rate numbers do not represent the ROI detection quality as there might be false-positive detection happened.

After the ROI was found, the cropped image would then be converted into greyscale to simplify three values representing RGB colour into only one number. The next step was to calculate the histogram of LBP values, and it ended with feature scaling for the sake of numerical stability.

This task uses SVM with a linear kernel as the classifier. During the hyperparameter tuning, the value of C was set to possible values of $\{1, 10, 100, 1{,}000\}$, with the $C = 1$ being the best value as the output from the grid search process. Fig. 6 shows the learning curve for task A1. The detailed explanation will be discussed in Section 5 of this report.

4.4. Task A2: Smile Detection Remarks

In this task, we were interested in the image of a lip to detect the smile. As depicted within Table 1, Dlib's 68-point facial feature detection was used for this matter. Out of 5,000 images, 169 of them were not able to be detected. Fig. 7 presented two samples of the image where the ROI detector failed to catch the region of lips. Unlike the majority of

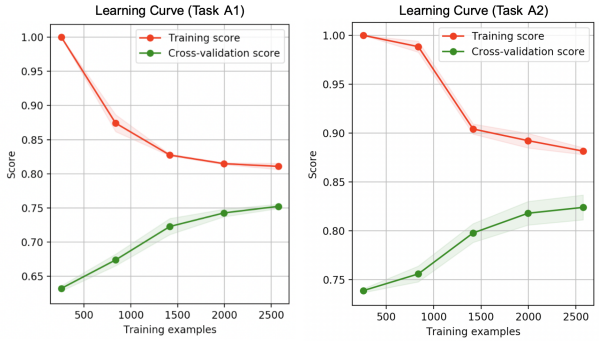

Fig. 6. The Learning curve of task A1 (left) and task A2 (right).

images within the dataset, these two images were not front-facing, which caused difficulty for the Dlib to mark the 68 points of facial features that consist of lip, jawline, eyes, and nose.

The entire grid search and cross-validation process took 4 hours to seek the best hyperparameter for the SVM linear. It was found that the value of $C = 1$ was the best hyperparameter for the given data.

Fig. 7. Two sample images where the area of interest was failed to be found by the ROI detector. Out of 5000 images, 97% of them were detected, while the rest, which mainly consisted of non-front-facing images, were not.

4.5. Task B1: Face Shape Classification Remarks

Fig. 8 shows that as the quantity of the training examples was sufficient, the cross-validation accuracy increased. Along with that, the training score adapted itself to be gradually lower and lower, which ultimately ended in having a low gap between training and cross-validation score. This is a good sign that the issue of overfitting might not be prevalent here [27]. The use of regularisation might contribute to this achievement. The nearly similar accuracy score between training score and validation score indicates the fact that the classifier is actually able to learn the underlying patterns and not merely memorising the data [27].

The best value of hyperparameter C for this task was 100. The whole hyperparameter tuning and cross-validation process took 39 hours to complete.

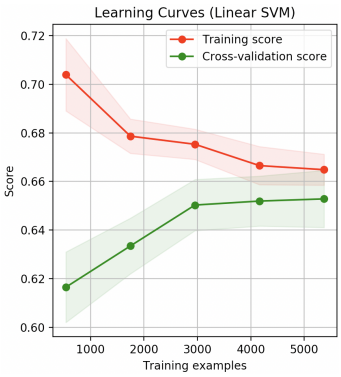

Fig. 8. B1 learning curve

4.6. Task B2: Eye Colour Classification Remarks

While all the tasks used Dlib's facial landmark detector, task B2 is an exception, where it used Haar-like features. To crop the eye region from the face, we used pre-trained Haar cascade by Shameem Hameed [28] that were trained manually tagging 20x20 pixels of frontal face images taken from Google Image. We used Haar-like, which performs well on detecting images that contain sunglasses as those were outliers for this task. Figure 9 presented the samples of glasses within the dataset.

Fig. 9. Some variance of glasses within the dataset. Transparent glasses, as depicted in the image (i) is acceptable as the eye colour is apparent. On the other hand, image (ii) is considered to be noise as the region of interest is not clearly visible

If we take a look at Fig. 10 on the cropping process, the resulted iris eye is visible, and no skin colour. The appearance of the skin area, which its values are varied across the images, might interfere with the classification process. There might be an area where we are not interested, such as the pupil and the sclera (the white outer layer of the eye). However, these appear and has nearly the same colour intensity across all of the images, making the signal to the classifier that those group of colours intensity are not necessary, perhaps similar to the concept of stopwords in computational linguistics [29].

The task B2 required features that sense the colour intensity, so unlike all the rest of the previous task, we could not use LBP. The raw pixels was used instead. This task required no less than 1,234 minutes to finish the grid search and cross-validation process. The best value of C gained was 1. The learning curve of this task is shown in Fig. 11, which will be discussed later in Section 5.

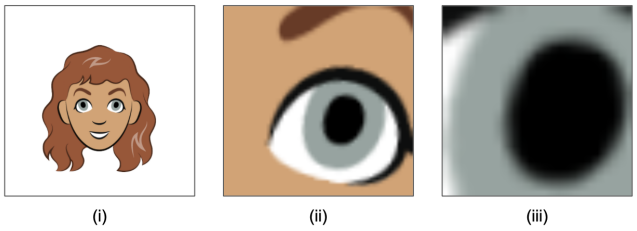

Fig. 10. The resulted images from the cropping process. Image (i) is the original file, image (ii) is the output from 68 shape detector. Image (iii) is the result of further cropping process to eliminate noise to this eye colour detection, such as skin colour and the colour of glasses frame if any.

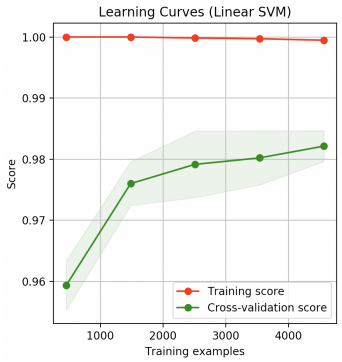

Fig. 11. B2 Learning Curve

5. EXPERIMENTAL RESULTS AND ANALYSIS

Task	Train Acc	Val Acc	Test Acc	Extra Test Acc
A1	81%	75%	75%	75%
A2	88%	82%	83%	86%
B1	66%	65%	67%	68%
B2	100%	98%	98%	98%

Table 2. Training, validation and testing score (accuracy) of each task

Table 2 shows the performance of each task, reported in the accuracy scores. The scores consist of training, validation, test, and an extra test set reported in accuracy percentage. Task B2 (5-class eye colour classification) gained the

best accuracy relative to others. This is mainly caused by the objective of task B2 is clearly defined within the RGB colour intensity features, making it the most straightforward task from the rest. Task B1 (5-class face shape classification) on the other hand, gained the lowest score, as it is the most laborious task to craft the features and hard to detect the region of interest, which is the coordinates of jawlines within an image.

Generally, there is no large accuracy gap between the original test set consists of 20% of the entire dataset, and the additional 1,000 images test set. Small accuracy gaps might happen due to several factors, such as the number of noise data within the test set, and the rate of false-positive and false-negative prediction that might vary and impacting the resultant accuracy.

Learning curve graph as the output of the training process can help us understand the underlying performance of a supervised classifier. As we refer back to Fig. 6, which shows task A1 and A2 learning curves, shows that as more training examples given, the gap between the training score and cross-validation score gets narrower. As depicted in Table 2, both of these two tasks has 0.60 score gap between training and validation score in the end. The smaller the gap between training and validation score, the better, as we can confirm that the model does not encounter with the issue of overfitting.

As for task B1, the small gap between training and cross-validation score in Fig. 8, especially at the rightmost area of the graph, it might indicate the absence of overfitting issue in learning the pattern of the dataset. A similar score between the two lines and the low score it gained, however, might indicate the presence of underfitting or high-bias problem. Some solutions to this are trying different kernels of SVM, or even try other classifiers that can fit the data better.

Task B2 gained the best accuracy score relative to other tasks presented in the report. The 0.2 difference between training and validation score is small relative to others, provided that it achieves its maximum score during the training process. This task is considered to be the simplest amongst the four. This was because the objective to guess the eye colour, clearly embedded as the features itself, which was the RGB pixel colour intensity.

6. CONCLUSION

The work in this report consists of four applied machine learning tasks, particularly within the area of computer vision. These tasks include two binary classification tasks: gender classification and smile detection, and another two multiclass classification problems: face shape and hair colour classification.

The process of extracting features is essential in these tasks, as these will be the input for the classifier. The feature extraction pipeline itself consists of mainly two series of steps: detecting the region of interest (ROI) within an image, then followed by the feature engineering process. For the first step, Dlib's facial landmark detection works very well on detecting the ROI. In a case where it did not work as expected, such as at task B2 where it falsely marked sunglasses as the region of the iris, Haar-like feature is being used instead. The detected ROI is then passed into some feature engineering techniques such as local binary pattern (LBP) histogram, coordinates of particular ROI edges, or even the raw RGB values per pixel.

As a classifier, linear SVM gives consistent performance reported by the accuracy scores. SVM with the linear kernel is also the most computationally inexpensive relative to other kernels for SVM. During the training process, it is also crucial to making use of grid search and cross-validation techniques to seek the best hyperparameter for the classifier and to validate the performance stability of the selected model.

As a result, the machine learning pipeline configuration for each task resulted in the variation of performance, ranging from 67% for task B1 to 98% for task B2. From the analysis, it is also apparent that reading the learning curves as the output of the training process can help us in diagnosing how the model performs. Analysing the learning curve can give us insight into the classical problem of underfitting and overfitting in machine learning. The learning curve can also give a hint on potential improvement for each task.

As for the future suggestion, it might be a good idea to try out another non-linear kernels for SVM to boost the accuracy score. It might also worth considering to try out deep learning-based approaches. These two suggestions are mainly related to the tasks that suffer from underfitting, such as task B1 (face shape classification).

7. REFERENCES

[1] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang, "Deep learning face attributes in the wild," *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2015 Inter, pp. 3730–3738, 2015.

[2] Google, "Cartoon Set: An Image Dataset of Random Cartoons," https://google.github.io/cartoonset/.

[3] P Viola and M Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, 12 2001, vol. 1, pp. I–I.

[4] Phillip Wilson and John Fernandez, "Facial feature detection using Haar classifiers," *Journal of Computing Sciences in Colleges*, vol. 21, no. 4, pp. 127–133, 2006.

[5] Sang Hyuk Lee, Jung Hawn Kim, Yong Jin Lim, and Joonhong Lim, "Traffic light detection and recognition based on Haar-like features," *International Conference*

on Electronics, Information and Communication, ICEIC 2018, vol. 2018-Janua, pp. 1–4, 2018.

[6] Gregory Shakhnarovich, Paul A. Viola, and Baback Moghaddam, "A unified learning framework for real time face detection and classification," *Proceedings - 5th IEEE International Conference on Automatic Face Gesture Recognition, FGR 2002*, pp. 16–23, 2002.

[7] Vahid Kazemi and Josephine Sullivan, "One Millisecond Face Alignment with an Ensemble of Regression Trees," 2014.

[8] Kivanc Yuksel, Xin Chang, and Władysław Skarbek, "Smile detectors correlation," *Photonics Applications in Astronomy, Communications, Industry, and High Energy Physics Experiments 2017*, vol. 10445, no. August 2017, pp. 104451L, 2017.

[9] Davis E King, "Dlib-ml: A Machine Learning Toolkit," *Journal of Machine Learning Research*, vol. 10, pp. 1755–1758, 2009.

[10] Choon Boon Ng, Yong Haur Tay, and Bok Min Goi, "A review of facial gender recognition," *Pattern Analysis and Applications*, vol. 18, no. 4, pp. 739–755, 2015.

[11] Timo Ojala, Matti Pietikäinen, and Topi Mäenpää, "Gray scale and rotation invariant texture classification with local binary patterns," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 1842, pp. 404–420, 2000.

[12] Hui Cheng Lian and Bao Liang Lu, "Multi-view gender classification using multi-resolution local binary patterns and support vector machines," *International Journal of Neural Systems*, vol. 17, no. 6, pp. 479–487, 2007.

[13] Luís A. Alexandre, "Gender recognition: A multiscale decision fusion approach," *Pattern Recognition Letters*, vol. 31, no. 11, pp. 1422–1427, 2010.

[14] F Pedregosa, G Varoquaux, A Gramfort, V Michel, B Thirion, O Grisel, M Blondel, P Prettenhofer, R Weiss, V Dubourg, J Vanderplas, A Passos, D Cournapeau, M Brucher, M Perrot, and E Duchesnay, "Scikit-learn: Machine Learning in {P}ython," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[15] Shai Shalev-Shwartz and Shai Ben-David, *Understanding machine learning: From theory to algorithms*, vol. 9781107057, 2013.

[16] Enrique Ramón-Balmaseda, Javier Lorenzo-Navarro, and Modesto Castrillón-Santana, "Gender classification in large databases," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7441 LNCS, pp. 74–81, 2012.

[17] Abdenour Hadid, Juha Ylioinas, Messaoud Bengherabi, Mohammad Ghahramani, and Abdelmalik Taleb-Ahmed, "Gender and texture classification: A comparative analysis using 13 variants of local binary patterns," *Pattern Recognition Letters*, vol. 68, pp. 231–238, 2015.

[18] UCL Information Services Division, "What is SSH and how do I use it?," https://www.ucl.ac.uk/isd/services/research-it/research-data-services/data-storage-service/storage-access-guide/what-ssh-and.

[19] Google, "Welcome to Colaboratory," https://colab.research.google.com/.

[20] Asa Ben-Hur and Jason Weston, "A user's guide to support vector machines.," *Methods in molecular biology (Clifton, N.J.)*, vol. 609, pp. 223–239, 2010.

[21] Andrew Ng, "CS229 Lecture notes Deep Learning," pp. 1–30, 2000.

[22] John D Hunter, "Matplotlib: A 2D graphics environment," *Computing in science & engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[23] the scikit-image contributors, "3.1. Cross-validation: evaluating estimator performance," https://scikit-learn.org/stable/modules/cross_validation.html.

[24] Stéfan van der Walt, Johannes L Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D Warner, Neil Yager, Emmanuelle Gouillart, Tony Yu, and the scikit-image contributors, "scikit-image: image processing in {P}ython," *PeerJ*, vol. 2, pp. e453, 2014.

[25] G Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[26] Travis E Oliphant, *A guide to NumPy*, vol. 1, Trelgol Publishing USA, 2006.

[27] Andrew Ng, "Diagnosing Bias vs. Variance," https://www.coursera.org/lecture/machine-learning/diagnosing-bias-vs-variance-yCAup.

[28] Shameem Hameed, "HAAR-cascade for eyes," http://www-personal.umich.edu/~shameem/haarcascade_eye.html, 2008.

[29] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze, "An Introduction to Information Retrieval," in *An Introduction to Information Retrieval*, p. 27. Cambridge University Press, 2009.