Face Shape Classification based on Machine Learning and Deep Learning Algorithms

Etinosa Enobun, P2681490
Mohammed Albardawil, P2661204

Problem Statement

Choosing the right eyelashes, hairstyle, and facial makeup has been always a vital process for most of women and young ladies. According to [1], it is always expensive and time consuming when it comes to measuring face characteristics, especially by experts. In addition, many eyewear shops tend to switch from being presented physically to have an online e-commerce store, and they employed the virtual try-on using Augmented Reality (AR) in which most customers can try the glasses virtually before placing orders and without the need to go to a physical store. The motivation of having an online store was due to the spread of COVID-19 virus all around the world. However, the customers still do not know which glasses suit the shape of their faces. Hence, face shape classification methods were introduced in this report. The project proposed 2 machine learning and 2 deep learning algorithms for image classification, and a dataset of female face shape images were preprocessed for classification training and testing purposes.

Aim

The project's aim is to classify female facial shapes using deep learning and machine learning methods. The various algorithms are implemented and compared to investigate and assess the various models to recommend the algorithm with the best accuracy and the training parameters to achieve it.

Objectives

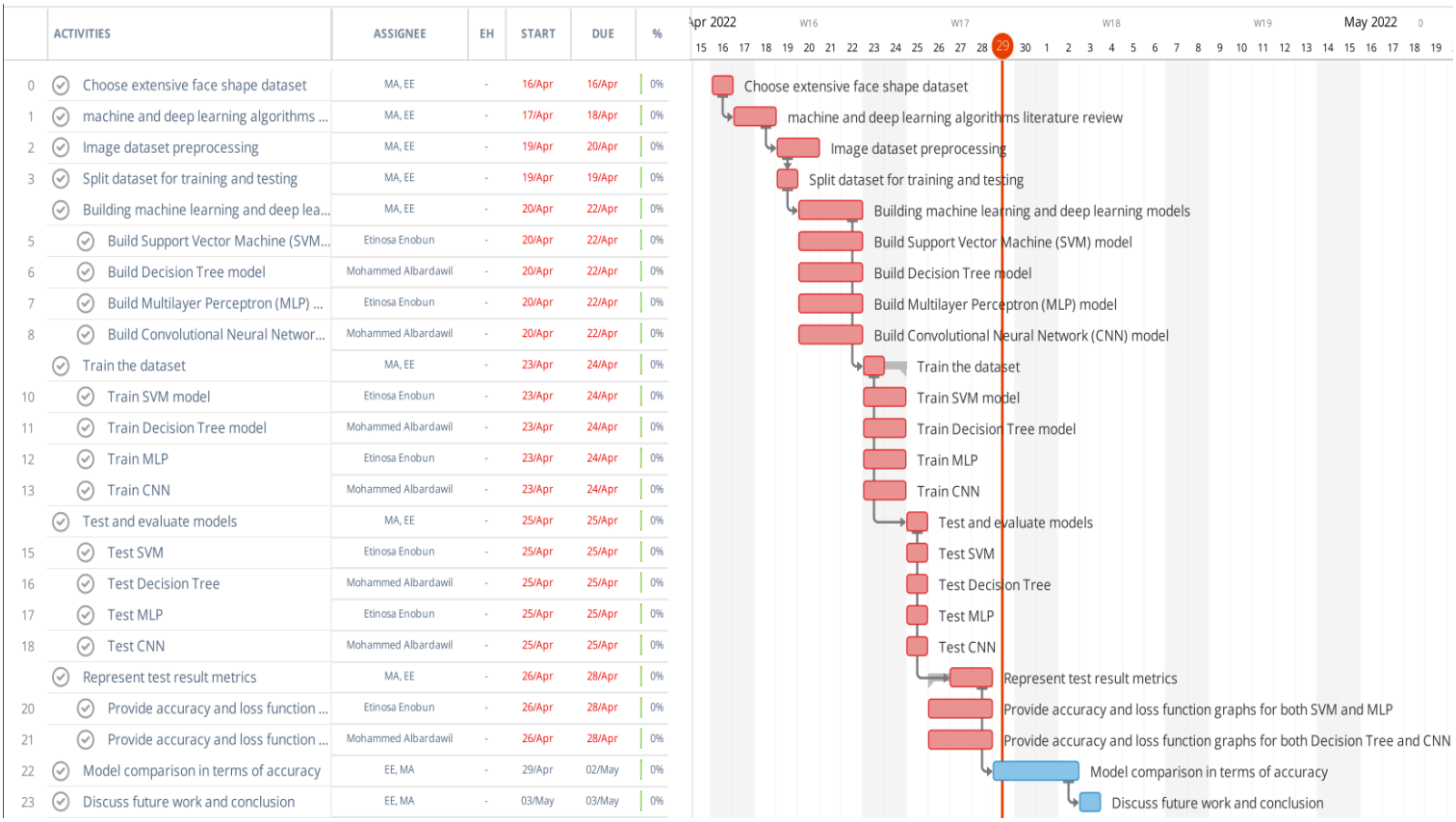
1. Find a good and extensive face shape dataset to be trained, tested, and evaluated using machine and deep learning models
2. Research about the most used machine learning and deep learning algorithms to build models for face shape classification
3. Preprocess the image dataset using image processing algorithms such as load and resize images so that all images have the same resolution to make them ready for the classification stage of the implementation
4. Split the dataset into training and validation dataset with 70% and 30%, respectively.
5. Building machine learning and deep learning models based on research literature review from Objective 2
6. Train the dataset using the training part of the dataset
7. Test and evaluate the model using the testing part of the dataset
8. Provide test results and represent them in metrics such as accuracy

9. Compare between the models to check which model has the highest accuracy
10. Discuss future work

Ethics Consideration

Several institutions and businesses have recently removed internet data sets containing thousands — if not millions — of images of people's faces that were used to enhance facial-recognition algorithms. These photographs were mostly taken from the internet by researchers. The images are considered public data, and their acquisition did not appear to raise any red flags with institutional review boards (IRBs) or other research-ethics agencies. However, none of the persons in the photos were contacted for permission, and some were offended by the manner their images were exploited [2]. The importance of public participation cannot be overstated; scientists should consult those whose lives are being described by the data. There are already some ethical guidelines for AI. Moreover, several US and European investors have backed attempts to investigate the difficulties of biometrics research and suggested redefining what constitutes "public" data, as well as asking academics to think about the study's potential harm to society. Due to the nature of this research, the dataset utilized had to be licensed and approved because this research required multiple photographs of persons that are considered personal data. The dataset used in [3] is licensed under the CC0: Public Domain as stated in [4]

Gantt Chart



Literature Review

Authors in [5] proposed face shape image classification novel algorithm based on Deep Convolutional Neural Networks (DCNN). This method was implemented to resolve the issue of measuring the characteristics of faces by beauty experts which is in fact costs time, money, and effort. The motivation behind this algorithm was to choose the right eyelashes, facial makeup, and of course the suitable eyeglasses or sunglasses especially for the eyewear store if they are going completely to be virtual. Because customers can not just try the glasses virtually, the algorithm may also recommend the best glasses based on history of customer using recommender systems. The designed algorithm combined the trained features using CNN with hand crafted features extracted using the Histogram Oriented Gradients (HOG) showed increased efficiency in face shape classification. The research's results proved that the accuracy of the proposed algorithm reached 81.1%.

Authors in [6] proposed an efficient face shape classification algorithm which decreases facial variations and occlusions of faces using Support Vector Machine (SVM) and Natural Features. Active Shape Model (ASM) was employed for natural features extraction, and features were categorized and then marked with their corresponding labels. A database of 4000 colorful images of face for more than 100 people was for training and testing purposes. Regarding experimentation, the resolution of all images was set to 60*80 pixels. The proposed method outperformed Principal Component Analysis (PCA), Latent Dirichlet Allocation (LDA) and the K-Nearest Neighbors (KNN) with accuracy of 89.6%.

Authors in [7] proposed a machine learning model that classified 4 types of Gerbera flowers based on their colour components in the center of flower and inside their petals. The whole workflow included several steps. The first step was building a database based on cropped each flower from images with the tag of their subtype. The second step was also preprocessing in which images' background was removed using filters and Hue color space transformation. Then, the highest value in histogram is identified to check which colour corresponds to which flower. The Decision Tree was applied on 70% of the data for training, and the rest were kept for testing and evaluation purposes. The proposed algorithm outperformed all other algorithms with 86.2% of accuracy.

Authors in [8] proposed a system that predicted the classes of new images by applying Multilayer perceptron with different training algorithms to a given dataset of images with known classifications. Various training algorithms were used to assess the proposed system's accuracy while maintaining the size and learning parameters constant. This project was designed as a classification system using a multilayer perceptron network, and it was evaluated using a variety of techniques, including Gradient Descent Backpropagation('traingd'), Gradient Descent with Adaptive Learning Rate Backpropagation ('traingda'), Resilient Backpropagation ('trainrp') amongst others. The findings reveal that some MLP (Multilayer Perceptron) training methods can provide as good a solution as other network topologies, with the Levenberg Marquardt ('trainlm') approach providing the best performance (78% accuracy) for MLP on image classification data.

The author in [9] proposed a system to improve the classification performance of support vector techniques by exploiting the spatial pixel association (SPA) characteristics extracted from hyperspectral data. The system was tested and experimented on the AVIRIS hyperspectral data over Indian Pine Site (IPS) to compare the performance of the classification approaches against some existing SVM based techniques such as SC-SVM and PSO-SVM and traditional methods like KNN AND K-means. The results

from the experimental tests showed that the proposed system outperforms the known classification algorithms, the SPA-SVM obtained an accuracy of 95.71%.

The authors in [10] proposed a system that uses SVM technique for classifying multispectral satellite image dataset and proceeded to compare the accuracy with conventional image classification method. The author highlighted that SVM is a kernel-based algorithm that builds a model for transforming low dimension feature space into high dimension feature space to find the maximum margin between the classes. The experiment carried was compared with image classification methods such as maximum likelihood, K-Nearest, which proved SVM to be one of the powerful kernel-based classifiers. Information was extracted from remote sensing data, which served as the source of the data which the algorithm was implemented on. Six classes were considered for the classification accuracy analysis. They are agricultural land, forest, water, sand, urban and rock. From the experiments, it showed that the SVM and MXL had the similar computation time, however the accuracy of SVM (92%) was higher than MXL (88%).

References

1. Alzahrani, T., Al-Nuaimy, W. and Al-Bander, B., 2021. Integrated Multi-Model Face Shape and Eye Attributes Identification for Hair Style and Eyelashes Recommendation. *Computation*, 9(5), p.54. Facial-recognition research needs an ethical reckoning. (2020). Nature, 587(7834), pp.330–330.
2. www.kaggle.com. (n.d.). Face Shape Dataset. [online] Available at: https://www.kaggle.com/datasets/niten19/face-shape-dataset [Accessed 29 Apr. 2022].
3. creativecommons.org. (n.d.). Creative Commons Legal Code. [online] Available at: https://creativecommons.org/publicdomain/zero/1.0/legalcode.
4. Alzahrani, T., Al-Nuaimy, W. & Al-Bander, B., 2019. Hybrid feature learning and engineering-based approach for face shape classification. 2019 International Conference on Intelligent Systems and Advanced Computing Sciences (ISACS).
5. Luo, L. et al., 2016. Face classification based on natural features and decision tree. 2016 International Conference on Virtual Reality and Visualization (ICVRV).
6. Vera Vera, J.E. et al., 2019. Classification of gerbera type flowers based in decision tree rules. 2019 XXII Symposium on Image, Signal Processing and Artificial Vision (STSIVA).
7. Coskun, N. & Yildirim, T., The effects of training algorithms in MLP network on Image Classification. Proceedings of the International Joint Conference on Neural Networks, 2003.
8. Baassou, B., He, M. & Mei, S., 2013. An accurate SVM-based classification approach for Hyperspectral Image Classification. 2013 21st International Conference on Geoinformatics.
9. Manthira Moorthi, S. et al., 2011. Kernel based learning approach for satellite image classification using support Vector Machine. 2011 IEEE Recent Advances in Intelligent Computational Systems.

Comparison between Convolutional Neural Networks and Decision Trees for Face Shape Classification

Mohammed Albardawil
Intelligent System and Robotics
DeMontfort University
Leicester, England
P2661204

***Abstract*—In this individual report, a Convolutional Neural Networks (CNN)-based model is implemented for face shape classification alongside Decision Trees (DTs), and then a comparison was made between both algorithms. The classification ran on models was based on females face shapes dataset imported from Kaggle website. Prior to model building and training, the data (images in this case) was preprocessed to ensure that models are only fed with face shape images of the same resolution. Several experiments conducted on models and the results proved that CNN outperformed the DT with an accuracy of 87.75%.**

Keywords—Artificial Intelligence, Computational Intelligence, Machine Learning, Deep Learning, Image Processing, Computer Vision

I. INTRODUCTION

Face shape classification has been a vital process for ladies to ease choosing the right makeup style, this includes hairstyle and eyelashes. In addition, eyewear shops tend to virtualize their stores and started to have online ecommerce shops to expand all around the world, equipping the virtual try-on technology that let people to try the glasses before placing their orders. However, customers would still go to physical eyewear stores to ask salespeople about which glasses best fit their faces, and unfortunately the ecommerce for makeup and eyewear still lacks the power of recommending which eyewear, eyelash, or even hairstyle that best fit their faces. In this paper, a CNN and DT based approaches have been used to implement a face shape classification system that detects shape of a female face shape based on the Kaggle's female face shape image dataset. Although this is not a recommender system, it is good to have small system that can at least predict the shape of faces in an offline mode, meaning that just to check if the input image has the correct image label based on the comparison between the input and the output image. In the next sections of this paper, algorithms, experimental design, and results and discussion are explained in detail.

II. ALGORITHM AND TECHNIQUE

Several methods have been created to classify data of different types, especially if the data was text-based and images-based. As soon as the images are the targeted data in this project, one machine learning and one deep learning algorithms have been implemented and a comparison made between both algorithms to verify which algorithm is best used in image classification applications, especially if the image data was complex, e.g., human face. Decision Trees (DT) and Convolutional neural Networks (CNN)

A. Convolutional Neural Networks (CNN)

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning classification method that takes an image as an input, assigns relevance to multiple aspects/objects in the image, and distinguish between them. When compared to other classification methods, the amount of pre-processing required by a ConvNet is significantly less. While basic approaches require hand-engineering of filters, ConvNets can learn these filters/characteristics with sufficient training [7]. The architecture of a ConvNet is inspired by the organisation of the Visual Cortex and is akin to the connectivity pattern of Neurons in the Human Brain. Individual neurons can only respond to stimuli in a tightened area of the visual field called the Receptive Field. A number of similar fields can be stacked on top of each other to span the full visual field [7].

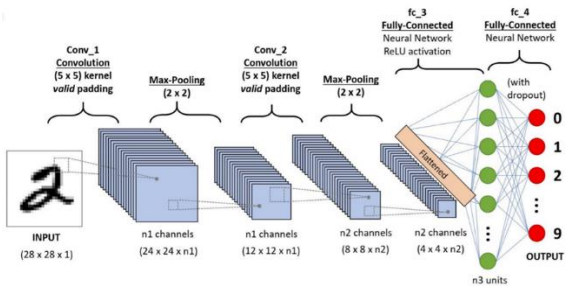


Figure 1: Overview of the CNN Architecture [7]

When it comes to preference between CNN and feed-forward, especially the Multilayer Perceptron (MLP), in image classification applications, CNN was the best choice. Why not to use MLP instead? As soon as the image is a 2D matrix data and can be easily flatten by transforming the 3x3 matrix dimension into 9x1, making it ready to be classified using MLPs. The answer to that question is the MLP can be a good choice only if the images were extremely binary basic, otherwise, MLP will not result in high accuracy [7]. Through the application of suitable filters, a ConvNet may successfully capture the Spatial and Temporal dependencies in a picture. Due to the reduced number of parameters involved and the reusability of weights, the architecture performs superior fitting to the picture dataset. In other words, the network may be trained to better recognize the image's sophistication. So basically, CNN is a collection of convolutional and pooling layers that enable the extraction of key features from images that best match the desired outcome. The most basic operations in CNNs is Padding. Padding means that all numbers in the matrix of a 2D image are edged (padded) with zeros to overcome the issue in

which only the middle elements of the matrix are used and the elements at the edge of the matrix are ignored [8]. Below is a representation of the padding operation:

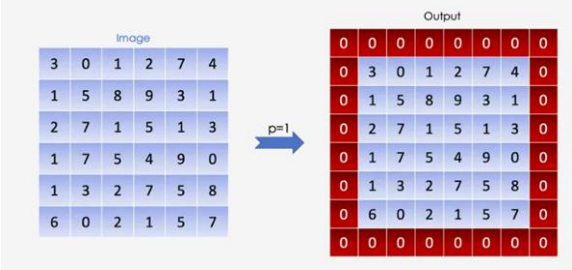


Figure 2: Padding of a grey-scale image [8]

After padding operation, the image is represented as a tensor, which is basically a mathematical model that can be used to describe physical features, as seen in the following equation:

$$\dim(image) = (n_H, n_W, n_C)$$

in which n_H is the height, n_W is the width and the n_C is the number of channels. In case of RGB image, the number of channels is 3 and the height and width depend on the image's resolution. In this project, the chosen resolution was 250x250 and it proved a high accuracy which is explained in detail in the testing and results chapter. Then, the dimension of the filter is defined as follows:

$$\dim(\text{filter}) = (k, k, nC), \text{ where k is the kernel}$$

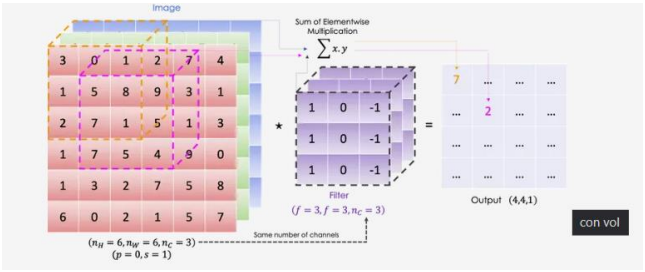


Figure 3: Representation of the convolution between an image and a filter [8]

For a given image I and filter K, the convolution would be:

$$conv(I,K)_{x,y} = \sum_{i=1}^{n_H} \sum_{j=1}^{n_W} \sum_{k=1}^{n_C} K_{i,j,k} I_{x+i-1, y+j-1, k}$$

As the "conv" function is considered as the filter in this case, its dimension becomes as follows:

$$dim(conv(I,K)) = (\lfloor \frac{n_H + 2p - f}{s} + 1 \rfloor, \lfloor \frac{n_W + 2p - f}{s} + 1 \rfloor); s > 0$$
$$= (n_H + 2p - f, nW + 2p - f); s = 0$$

Where: $\lfloor x \rfloor$ is the floor of x, p is Padding, s is the Stride, and since the stride in this project is zero, then the second equation is applied. Another essential layer in the network topology that comes after padding, is the Pooling. Pooling's main function is to downsample the features of a picture by adding up all of its information through each channel, keeping the n_C remain the same and changing both n_H *and* n_W.

$$dim(conv(I,K)) = (\lfloor \frac{n_H + 2p - f}{s} + 1 \rfloor, \lfloor \frac{n_W + 2p - f}{s} + 1 \rfloor); s > 0$$
$$= (nH + 2p - f, nW + 2p - f); s = 0$$

Although there is an Average Pooling, the pooling applied in this project is the MaxPooling, in which the maximum of all elements in the filter is returned. Now, all of the aforementioned operations are combined to form the so-called CNN network topology. Initially, there are three main layers. The first layer is called the Convolutional Layer followed with activation function. The second layer is the Pooling Layer and the last one is the Fully Connected Layer, which was brought by the Feed Forward-Neural Networks. In order to define the Conv Layer, the following parameters have to be set first:

- Input Size (Image input): $a^{[l-1]}$ with size $(nH^{[l-1]}, nW^{[l-1]}, nC^{[l-1]})$
- Padding: $p^{[l]}$
- Number of Filters: $nC^{[l]}$
- Activation Function: $\psi^{[l]}$
- Output Layer: $a^{[l]}$ with size $(nH^{[l]}, nW^{[l]}, nC^{[l]})$

Thus, the final equation and its dimension is as follows:

Type equation here.

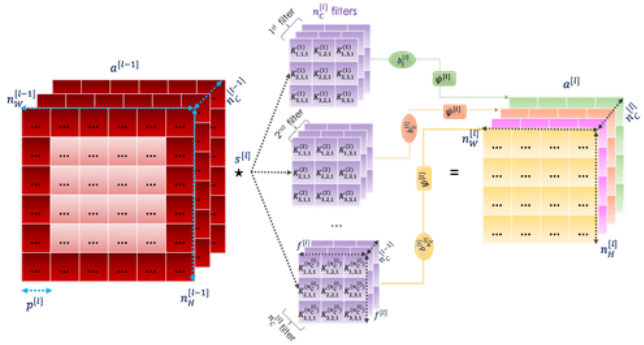


Figure 4: Illustration of CNN's First Layer [8]

The same parameters applied to Pooling except that instead of having activation function, it has pooling function denoted as $\phi^{[l]}$.

$$pool(a[l-1])x, y, z = \phi[l]\left((ax + i - 1, y + j - 1, z[l-1])(i,j) \in [1,2,\dots,f[l]]2\right)$$

After that, a fully connected layer is defined. Fully connected are basically a finite number of neurons, considering the following equation for j^{th} node and i^{th}layer:

$$zj[i] = \sum l = 1ni - 1wj, l[i]al[i-1] + bj[i] \text{ \rightarrow a^{[i]}_j =\psi^{[i]}(z^{[i]}_j)}$$
$$\rightarrow aj[i] = \psi[i](zj[i])$$

All of the aforementioned operations are then combined into a pipeline of neural networks.

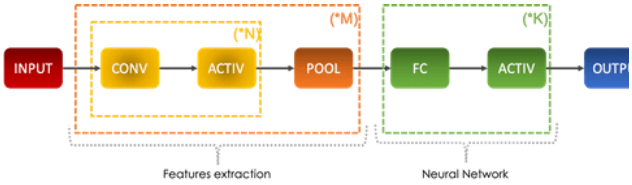


Figure 5: CNN Pipeline [8]

B. *Decision Trees*

According to [9], the Decision Trees algorithms is a basic yet effective supervised learning technique in which data points are continuously separated according to particular parameters and/or the problem that the algorithm is attempting to resolve. A decision tree constructs classification or regression models. It incrementally cuts down a dataset into smaller and smaller sections while also developing an associated decision tree. A root node, branches, and leaf nodes are the basic building blocks for every decision tree. The numerous test scenarios are described by the internal nodes within the tree. Both classification and regression issues can be solved with Decision Trees. The algorithm can be conceived of as a graphical tree-like structure that predicts outcomes using various tuning parameters. The dataset that is fed during training is processed top-down by the decision trees as stated in [9].

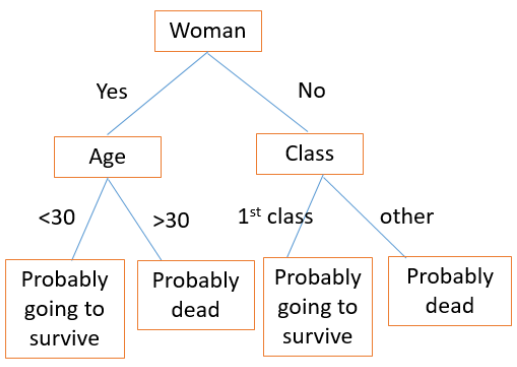


Figure 6: Decision Tree structure [9].

To predict the result of a problem, the Decision Tree method employs a data structure known as a tree. The algorithm is fed with a collection of pre-processed data because the decision tree uses a supervised method. The algorithm is trained using these data. The basic concept is to split the data space into dense and sparse sections using a decision tree. A binary tree can be split in two ways: binary or multiway. The method continues to split the tree until the data is homogeneous enough. A decision tree is returned at the end of the training that can be utilized to make optimal classified predictions.

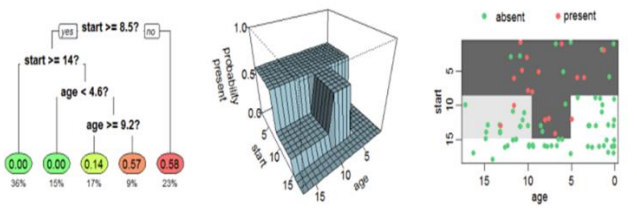


Figure 7: Representation of Decision Trees prediction with results [9]

Entropy is an essential term in the construction of this algorithm. It can be thought of as a measure of a dataset's uncertainty, and its value represents the degree of randomness of a given node. When the margin of difference for a result is very small, the model loses confidence in the accuracy of the prediction. The higher the entropy, the more randomness there will be in the dataset. A lower entropy should be preferable when creating a decision tree. The following is the formula for estimating the entropy of a decision tree:

$$Entropy = \sum_{i=1}^{c} - p_i * log_2(p_i)$$

There is just one root node in each tree. In comparison to all other features, the root node is frequently regarded as the most significant element. The root node is usually picked as the feature with the best accuracy among the others.

III. Experimental Design

The whole system was written in Python with the help of several libraries. Numpy library was mainly used for data structures and matrices manipulation, OpenCV (denoted as cv2 in Python) was used for preprocessing image dataset, and matplotlib was used to represent the evaluation of models in graphs. Regarding Machine Learning (ML) and Deep Learning (DL) algorithms, Tensorflow framework was extensively employed in this project for both algorithms. However, CNN algorithm was implemented using Keras, which is another layer of Tensorflow that is meant for Neural Networks (NN) and DL algorithms. The reason behind choosing Keras for DL models is that it made it easier for AI developers to implement algorithms with less, highly optimized code, especially that Keras being a Tensorflow layer for DL models, Tensorflow was already built in C/C++ which makes it more performant compared to other languages. This means that Decision Trees classifier was implemented using Tensorflow, and CNN classifier was written using Keras interface. At first, the experiments were conducted on a local machine using a CPU, however, the time taken to train models was more than two hours. Thanks to Google Colab, which made access to Google Jupyter Notebook, enabling developers to use high computational resources like GPUs which made the training for CNN model to finish in less than 10 minutes. There are three main stages for the whole system. Firstly, data preprocessing. Secondly, building, training, and testing models. Thirdly, models evaluation, which is described in the Testing and Results of this report.

A. Data Preprocessing

The dataset is filtered, cleared and ready to be processed and trained in this stage. By convention, all machine learning/deep learning algorithms are fed with training portion and evaluated based on the testing portion of the whole dataset. Conventionally, the dataset is split into 70% and 30% to training and testing, respectively. However, in this project the validation split was set to 0.2, meaning that the training data is 80% of the dataset and the rest was kept for testing ad evaluation. Regarding the dataset, the image shapes were classified into 5 different shapes: Oval, Square, Round, Oblong and Heart. The size of the dataset is around 5000, 4000 reserved for training and the rest was for testing.

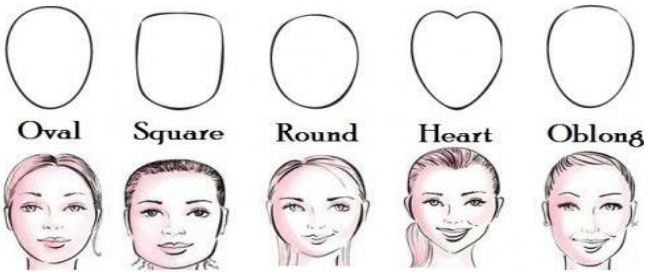


Figure 8: Types of Face Shapes [10]

All images have been resized to 250x250 using the function cv2.resize() from OpenCV library to in order to have fair training and testing processes on all images. In addition, prior to retrieving images for training and testing datasets, the images were randomly selected for both datasets.

B. Model Implementation

Both parts were implemented separately. However, both models were implemented to check which model is best for image classification by evaluating each part's accuracy.

1) Convolutional Neural Networks

Before getting started with the CNN, its design parameters have to be defined before the beginning of training. These parameters include number of layers, number of filters, kernel dimension, activation function, and the image shape resolution as well as number of channels. The number of layers is 3 because the CNN is always based on 3 or more layers, and if more than three layers chosen, it means there is more computational cost on the system. For this project, 3 layers are enough to avoid having more computational cost as well as reducing the chance to have overfitting, which may lead to lose important data during training and also leading to have a lower accuracy. As stated in [11], it is decent to have a kernel size of 3x3 to have less computation and easier to learn large complex features compared to 5x5 or even 7x7 kernel sizes. The same applies to the number of filters. Then, there were three most used activation functions for training an NN, which are Tangent Hyperbolic (tanh), sigmoid and Rectified Linear Unit (RELU). The chosen activation function is RELU because sigmoid and tanh functions are complex compared to the RELU in terms of calculation, only if the application is image classification related [12]. Otherwise, sigmoid and tanh are better options. Nonetheless, sigmoidal functions tend to reach 0 for high input values for neuron's gradient, referred to as "vanishing gradient", which is prevented by the RELU function.

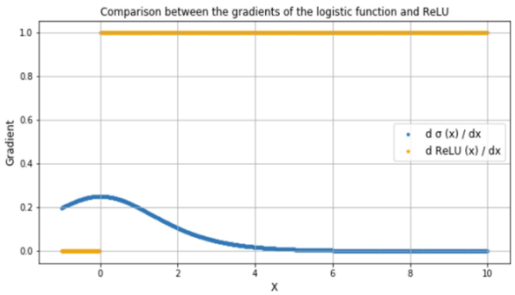


Figure 9: Comparison between ReLU and Signmoid's functions' gradient

Now, the image resolution was set to 250x250 as it was proved that each time the resolution size increases, the accuracy is also increased, however, it will result in more computational time. After that, there are some of the hyperparameters that need to be defined prior to training such as batch size, number of epochs and number of iterations. First of all, batch size means number of training samples that are run in one iteration as mentioned in [13]. Second of all, the number of epochs is basically one time processing for both backward and forward for a batch of data [14]. The most crucial hyper-parameter in NN training is the number of iterations, which is simply number of batches required to finish one epoch. Both size of epochs and batch size are already defined with 90 and 350 for CNN, respectively. Those values were set after some experiments and testing which is explained in the following chapter. Then, the number of iterations is obtained through the following equation:

$$iterations = \left(\frac{no.of\_epochs}{batch_{size}}\right) * no.of\_training\_samples$$

This means that the number of iterations based on the previous values is approximately 15556 iterations, as the number of training samples in this project was set 3999 images based on the 80% of the total dataset, and the rest were set to testing. The more the number of iterations, the higher the time and computational costs and the higher the accuracy of the NN model.

2) Decision Trees

As mentioned earlier in the Algorithms Used section, the Decision Tree classification is based entirely on based on the criterion, and the chosen criterion was the Entropy function. The entropy was used to calculate the dataset's uncertainty and its value represents the randomness of the algorithm. On the one hand, CNN was built using TensorFlow Keras. On the other hand, Decision Tree was built using the Scikit-Learn library as it made it easier and faster for developers to code machine learning algorithms without the need to spend time write the whole algorithm from scratch.

IV. Results and Discussion

There were experiments conducted on this study and a comparison made between all algorithms, both machine learning algorithms and both deep learning algorithms. Metrics have mentioned in this project for evaluation were two, the accuracy and the loss function for deep learning models, and only the accuracy for machine learning algorithms. Regarding DL algorithms, both the loss function and the accuracy were applied against the number of

iterations for both training and testing. Testing accuracy and loss function is always less than the training, and therefore, the focal point is the training accuracy and loss function. Below is a graph representation for the accuracy and loss function of CNN model, based on the following parameters (number of epochs: 75, batch size: 128, image resolution: 150x150, bringing 2344 iterations):

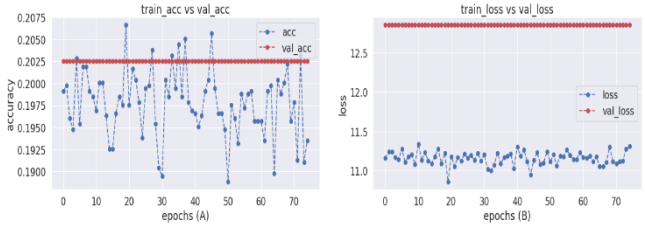


Figure 10: (A) Accuracy (acc) vs No. of Epochs, (B) Loss Function vs No. of Epochs (fourth attempt)

The average accuracy for both training and testing was 20% and the average loss function for both training and testing was 12.89. The time taken to train the data was 9 mins. The accuracy is too low for a model to be used later in AI applications. Thus, another training attempt has been run as another experiment based on the following hyperparameters (number of epochs: 100, batch size: 128, image resolution: 150x150, bringing 3125 iterations) to get the below results:

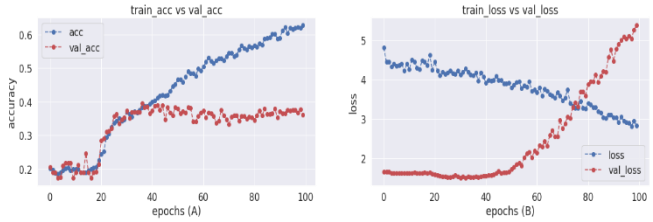


Figure 11: (A) Accuracy (acc) vs No. of Epochs, (B) Loss Function vs No. of Epochs (second attempt)

The accuracy was almost the same, reaching 72% for training, 1.5903 for loss function and calculated training time was 12 mins. Now, it seemed that there was a significant rise in the accuracy level. Yet the accuracy has to be improved. The third attempt performed had the same parameters except for the number of epochs, which was double the last attempt.

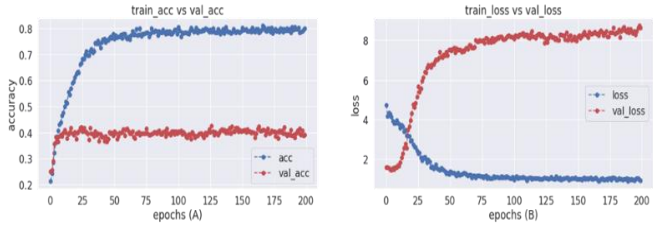


Figure 12: (A) Accuracy (acc) vs No. of Epochs, (B) Loss Function vs No. of Epochs (third attempt)

In this experiment, the training has reached a very good accuracy of 87.75% with loss function of a about 1.7. The time taken to finish the training in this experiment was 20 minutes. Thanks to Google Colab GPU resources, which made the training to finish 9 times faster compared to the training applied using CPU. One more experiment performed to check if there is a possibility to have more than 90% accuracy when the number of epochs increased to 350. However, this it was thought of increasing the resolution to become 250x250 and the reducing the batch size to 90 than 128 which might result in a higher accuracy. Unfortunately, this did not make any difference because it reached almost the same accuracy of 87.07%, loss function of 1.74 and the time taken 1 hour and 27 minutes, which is basically too slow to train a model while it can have the accuracy with 4.35 times faster.

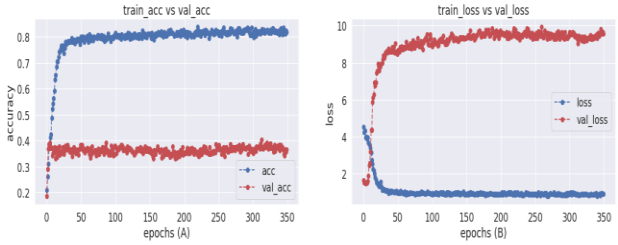


Figure 13: (A) Accuracy (acc) vs No. of Epochs, (B) Loss Function vs No. of Epochs (fourth attempt)

Regarding the Decision Trees, only one attempt was performed with 250x250 image resolution and using the entropy as the criterion, the accuracy of the classifier was almost 24.6% which is too low compared to the CNN classifier that reached around 87.75%. The accuracy was too low compared to the CNNs accuracy, reaching 24.6%. the reason behind this is that the CNNs were always meant for large 2-dimensional data, and Decision Trees classification can handle the one dimensional, e.g., texts and Natural Language Processing applications, because the classification is linear and are not able to process the non-linear problems.

V. Conclusion

This research presented an effective method for merging a neural network as CNN with a machine learning DT model. This model's goal is to categorise facial forms. The system is highly automated, with CNN and DT performance of around 88% and 27%, respectively. To increase its performance, it can be adjusted and combined with other categorization methods. The technique was successfully developed for facial shape classification.

References

[1] Medium. 2022. *A Comprehensive Guide to Convolutional Neural Networks—the ELI5 way*. [online] Available at: <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53> [Accessed 29 April 2022].

[2] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[3] I. S. Jacobs and C. P. Bean, "Fine particles, thin films and exchange anisotropy," in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.

[4] Chemoheadwear.co.uk. 2022. [online] Available at: <https://www.chemoheadwear.co.uk/identify-your-face-shape/> [Accessed 30 April 2022].

[5] Medium. 2022. A Comprehensive Guide to Convolutional Neural Networks—the ELI5 way. [online] Available at: <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53> [Accessed 29 April 2022].

[6] 2022. [online] Available at: <https://www.baeldung.com/cs/ml-relu-dropout-layers> [Accessed 29 April 2022].

[7] Gaillard, F., 2022. Batch size (machine learning) | Radiology Reference Article | Radiopaedia.org. [online] Radiopaedia.org. Available at: <https://radiopaedia.org/articles/batch-size-machine-learning?lang=gb#:~:text=Batch%20size%20is%20a%20term,iteration%20and%20epoch%20values%20equivalent> [Accessed 30 April 2022].

[8] Brownlee, J., 2022. Difference Between a Batch and an Epoch in a Neural Network. [online] Machine Learning Mastery. <https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/#:~:text=The%20number%20of%20epochs%20is%20the%20number%20of%20complete%20passes,value%20between%20one%20and%20infinity.> [Accessed 30 April 2022].

Face Shape Classification Using Multilayer Perceptron Algorithm and Support Vector Machine.

Abraham Etinosa Enobun
DeMontfort University
Leicester, England
P2681490

***Abstract*—This report describes the development of a face shape classifier implemented using one deep learning algorithm (Multilayer Perceptron) and One Machine Learning technique (Support Vector Machine). A comparative study is conducted on the performance of both algorithms when implemented. Certain hyperparameters in each model were varied until an optimal set of condition was reached. The performance of the models was compared based on metrics such as precision, accuracy, fl score and training time. The MLP outperformed the SVM algorithm with an accuracy of 87.5%.**

Keywords—deep learning, machine learning, MLP, SVM, Perceptron, face classification, computer vision, computational intelligence.

I. INTRODUCTION

The face shape can be categorised into five groups, according to beauty expert guidelines: round, oval, oblong, square, and heart forms. The primary issue is to apply different set of algorithms to distinguish the different categories of face shapes form a set of face images. This paper hopes to solve the major issue with determining the right shape of the human face. The dataset was obtained via Kaggle, and the images were processed, after which the algorithms were applied on the images and the performance are measured and compared.

II. ALGORITHM AND TECHNIQUE

In this section, the notation, the ANN architecture, and training process are explained. Neural networks have been widely used in pattern classification, completion, approximation, prediction, and optimizations.

A. MultiLayer Perceptron

Starting with the considerably simpler single layer perceptron before moving on to the multilayer perceptron. A perceptron is a single-neuron model that is a blueprint for larger neural networks. The perceptron accepts *n* inputs from diverse features *x* and creates an output with varied weights *w*. A perceptron is a linear classifier, which implies it is an algorithm that classifies data by drawing a straight line between two categories. A perceptron generates a linear combination using its input weights to produce a single output depending on numerous real-valued inputs (and sometimes passing the output through a nonlinear activation function). Here's how you can express it mathematically:

$$y = (\varphi \sum_{i=1}^{n} \omega_i x_i + b\ ) = \ \varphi(W^T \boldsymbol{X} + b)$$

where **w** denotes the vector of weights, **x** is the vector of inputs, **b** is the bias and $\boldsymbol{\varphi}$ is the non-linear activation function.

A fully connected multi-layer neural network is called a Multilayer Perceptron (MLP). The MLP is an ANN with a feed-forward design that comprises of many layers of neurons. A multilayer perceptron has three or more layers, each with an input, output, and one or more hidden layers. The neurons in MLP have a nonlinear activation function, and each layer is fully connected to the next. Using non-linear/linear activation functions, several perceptrons are merged to generate a decision boundary. A non-linear mapping to a new dimension is provided by each perceptron. Given that MLP is a fully connected network each neuron in each layer is connection to the next layer with a certain weight function w_{ij}. MLP uses a supervised learning technique called back propagation. The weight function of the neural network is defined during the training phase. The MLP's training method is based on the minimization of a cost function that was first proposed by Werbos [1] and Parker [2].

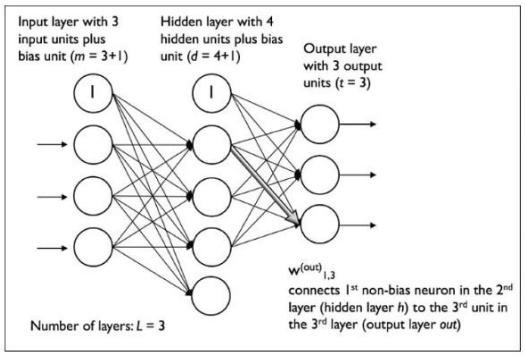


Figure 1: Multilayer Perceptron

Figure 1 consist of 3 layers: input layer, hidden layer, and the output layer. An MLP is a typical example of a feedforward artificial neural network. The weight adjustment training is done via backpropagation. There are two steps to the algorithm. The predicted outputs corresponding to the given inputs are evaluated in the forward pass. The partial derivatives of the cost function with respect to the various parameters are propagated back through the network in the backward pass.

In the forward pass, the signal flow flows from the input layer via the hidden layers to the output layer, and the output layer's decision is compared to the ground truth labels. In the backward pass, partial derivatives of the error function with respect to the various weights and biases are backpropagated through the MLP using backpropagation and the chain rule of calculus.

The act of differentiation creates a gradient, or error landscape, along which the parameters can be tweaked to bring the MLP closer to the error minimum. Any gradient-based optimization algorithm, such as stochastic gradient descent, can be used to accomplish this. This state is described as convergence. Data processing is improved by deeper neural networks. Deeper layers, on the other hand, can result in problems with vanishing gradients.

B. Support Vector Machine

Support vector machine (SVM) is a supervised machine learning model that can be used for classification and regression. However, it is mostly used in classification problems. In SVM algorithm, each data item is plotted as a point in n-dimensional space (where n is the number of features), with the value of each feature being the value of a certain coordinate in the SVM algorithm. Then we accomplish classification by identifying the hyper-plane that clearly distinguishes the two classes.

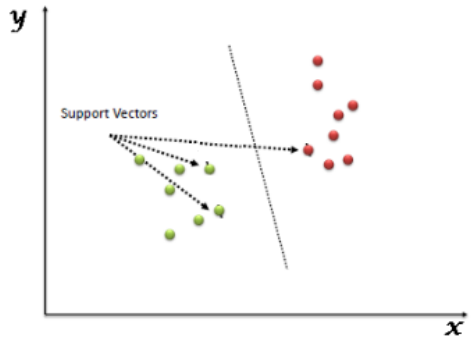


Figure 2: Support Vector Machine

The input to a SVM algorithm is a set $\{(x_i,y_i)\}$ of labelled training data where x_i is the data and y_i= -1 *or* 1 is the label. The output of a SVM algorithm is a set of N_s support vectors s_i , coefficient weights α_i, class labels y_i of the support vectors, and a constant term b. The linear decision surface is

$$w \cdot z + b = 0$$

Where,

$$w = \sum_{I=1}^{N_s} \alpha_s y_i s_i$$

An advantage of SVM is it can handle both continuous and categorical variables with ease. To differentiate various classes, SVM creates a hyperplane in multidimensional space. SVM iteratively generates the best hyperplane, which is then utilised to minimise an error. The goal of SVM is to find a maximum marginal hyperplane (MMH) that splits a dataset into classes as evenly as possible.

SVM also provides a method for linear non-separable data, which uses a technique known as "kernel mapping" to map data in input space to a high-dimensional feature space, where the data is subsequently linearized [3].

SVM can be extended to nonlinear decision surfaces by using a kernel K (·, ·) that satisfies Mercer's condition [6, 7]. The non-linear decision surface is

$$\sum_{I=1}^{N_s} \alpha_i y_i K(s_i, z) + b = 0$$

III. EXPERIMENTAL DESIGN / IMPLEMENTATION

Given the extremely advantageous Scikit-learn module, which is today regarded one of the most popular python libraries for machine learning, the SVM algorithm was implemented in Python. The dataset can be imported through the library, which provides access to pre-processing functions such as data normalisation, train and test image splitting, accuracy metrics, and confusion matrix, among other things. This will allow us to build a system using SVM classifiers, fine-tune it to get the best classification result.

For the MLP implementation, the Kera python library was adopted. Keras is a Python-based deep learning API that runs on top of the TensorFlow machine learning platform. It was created with the goal of allowing quick experimentation. Keras' primary data structures are layers and models. The Sequential model, which is a linear stack of layers, is the most basic sort of model. Keras functional API is used for more complicated designs, which allows users to build arbitrary layer graphs or write models entirely from scratch via subclassing [9].

A. Dataset (Image) Processing:

After the import of the dataset, the following were noted: The dataset has been split into two folders of training data and test data. Each folder containing folders of the class labels namely: Oblong, Round, Square, Heart, and Oval. There are 1000 images in each category. Each category's Training Set has 800 images, while the Testing Set contains 200 images.

The aim of this process is to improve image data (features) by suppressing undesired distortions and enhancing key image qualities so that the Computer Vision models can operate effectively with it.

Important pre-processing actions carried out on the images are:

1. Image resizing: the images were resized to (150 ,150).
2. Colour conversion of images: The images were opened with OpenCV, when we use cv2.imread() to open an image in OpenCV, it is shown in BGR format by default (BGR stands for Blue (255, 0, 0), Green (0, 255, 0), Red (0, 0, 255). It also has colour-changing tools for translating a BGR image into various Colour spaces using cv2.cvtColor(). The images are converted from BGR to RGB

The training images were split into training data and validation data with a ratio of 80:20. The validation dataset is to provide an unbiased evaluation of a model fit on the training dataset while tuning the hyperparameters [5]. The images were shuffled in preparation for loading the training data to eliminate any bias in the system.

B. System/Model Design

- *MLP Modelling*

The MLP algorithm was constructed as a sequential model to include 3 layers. The number of layers is three since the MLP is always based on three or more layers. Selecting more than three layers indicates the system will have a higher computational cost. To avoid overfitting and ensure no loss important data during training, it is sufficient to use 3 layers for the modelling.

In the first layer of the model, the input shape (image resolution) was defined as (150,150,3). It has been observed that as the resolution size grows, so does the accuracy. The layers were fully connected layers (dense). For the 3 layers, a dropout was applied to the model setting a fraction of inputs to zero to avoid overfitting.

In choosing the activation function, it was noted that in image classification, RELU is the preferred option. RELUs speeds

up the training, the gradient computation is very simple (either 0 or 1 depending on the sign of x), hence don't suffer from vanishing gradients, the computational step of a RELU is easy in comparison with sigmoid and tanh functions.

For the model compilation, the major attributes accepted includes the model optimizer, Loss function and metrics. The model optimizer is the technique used to update weights in the model, in this project, the Adam optimiser (Adaptive moment estimation) was selected. The Adam optimizer is a gradient descent optimizer uses adaptive learning rates. The model loss function which is also known as the objective function is the evaluation of the model used by the optimizer to navigate the weight space. The '*sparse_categorical_crossentropy'* was selected as it's for multiclass logarithmic loss. The model metric evaluated is the accuracy.

Following that, several hyperparameters, such as batch size, number of epochs, and number of iterations, must be defined prior to training. As defined in [12], batch size refers to the number of training instances shown to the model before a weight update is performed. For both backward and forward processing of a batch of data, the number of epochs is essentially one time processing. The number of iterations, or the number of batches required to complete one epoch, is an essential hyper-parameter in Neural Network training.

The MLP learning procedure:

The Multilayer Perceptron (MLP) learns in the following way:

1. Begin by propagating data from the input layer to the output layer. Forward propagation is the next phase.
2. Determine the error based on the output (the difference between the predicted and known outcome). The mistake must be kept to a minimum.
3. Backpropagate the error. Update the model by finding its derivative with respect to each weight in the network.

- *SVM Modelling*

SVM creates a hyperplane or a series of hyperplanes in a high-dimensional space, and the hyperplane with the greatest distance to the nearest training data point of each class achieves good separation between the two classes.

The kernel function that is employed determines the algorithm's true power. The following are the most widely utilised kernels:

1. Linear Kernel
2. Gaussian Kernel
3. Polynomial Kernel

Scikit-learn is a popular library for implementing machine learning algorithms in Python. SVM is also available in the scikit-learn library, and we use it in the same way (Import library, object creation, fitting model, and prediction).

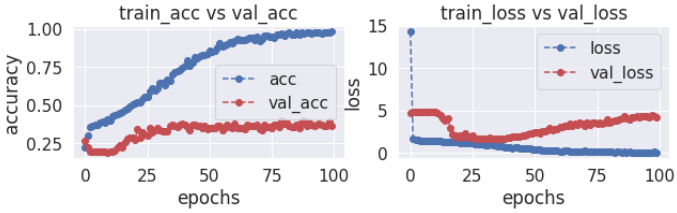
Important parameters varied to obtain the desired results are the kernel and gamma.

- Kernel: {'linear', 'poly', 'rbf', 'sigmoid', 'precomputed'} or callable, default='rbf'. Specifies the kernel type to be used in the algorithm. In this project the linear kernel was selected. Linear SVM kernel is used due to the large number of features (>1000), because the data is more likely to be linearly separable in high-dimensional space.
- Gamma: {'scale', 'auto'} or float, default='scale'. Kernel coefficient for 'rbf', 'poly' and 'sigmoid'. if 'auto', uses 1 / n_features, this was selected option for this project.

IV. EVALUATION AND RESULTS

The MLP system was adjusted to allow for the usage of the above-mentioned parameters, and it was utilised as a pre-cursor before applying the method. The algorithms were trained on the datasets. A comparative experiment study was carried out to understand the influence of the hyperparameters on the accuracy of the models. In this research, two metrics were measured for evaluation: accuracy and loss function for deep learning models, and merely accuracy for machine learning methods. Because the training accuracy and loss function is always less than the testing accuracy and loss function, the training accuracy and loss function is the main point for comparison.

*A. Based on the following parameters (Number of epochs:**100**, Batch size:**90**, image resolution: **150x150**), a graph representation of the accuracy and loss function of the MLP model is shown below.*

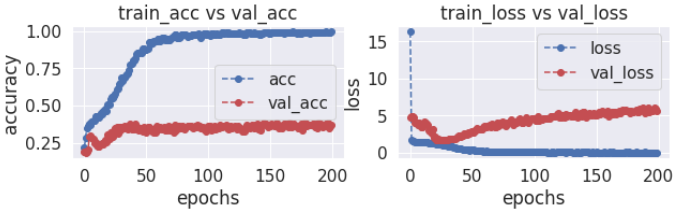


Training time: *47min 23s*
Training Accuracy: *87.1%*
Testing accuracy: *31.4%*

Classification Report (Testing)

	precision	recall	f1-score	support
Heart	0.30	0.28	0.29	200
Oblong	0.31	0.39	0.35	200
Oval	0.28	0.16	0.20	200
Round	0.40	0.37	0.39	200
Square	0.37	0.48	0.42	200

*B. Based on the following parameters (Number of epochs:**200**, Batch size:**90**, image resolution: **150x150**), a graph representation of the accuracy and loss function of the MLP model is shown below.*

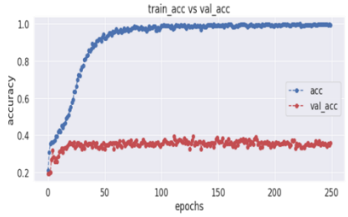


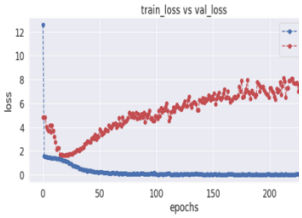
Training time: *1h 34min 8s*
Training Accuracy: *87.5%*
Testing accuracy: *32.1%*

Classification Report (Testing)

	precision	recall	f1-score	support
Heart	0.34	0.32	0.33	200
Oblong	0.33	0.44	0.38	200
Oval	0.22	0.18	0.20	200
Round	0.37	0.26	0.30	200
Square	0.34	0.42	0.38	200

C. Based on the following parameters (Number of epochs:250, Batch size:60, image resolution: 150x150), a graph representation of the accuracy and loss function of the MLP model is shown below.





Training time: *2h 5min 23s*
Training Accuracy: *87.0%*
Testing accuracy: *32.4%*

Classification Report (Testing)

	precision	recall	f1-score	support
Heart	0.34	0.32	0.33	200
Oblong	0.33	0.44	0.38	200
Oval	0.22	0.18	0.20	200
Round	0.37	0.26	0.30	200
Square	0.34	0.42	0.38	200

The SVM algorithm proved to be of very low accuracy, varying the type of kernel used complicated the training as this was taking too long to train on the data. The best performance by the algorithm was running on the linear kernel, and this measured to be 32%.

CONCLUSION

An efficient approach combining a neural network as MLP and a machine learning SVM model is presented in this paper. The aim for this model is to classify facial shapes. The system is highly automated, with MLP and SVM performance of approximately 90 % and %, respectively. It can also be tweaked and coupled with other classification systems to improve its performance. The technique was successfully built for the classification of facial shapes.

REFERENCES

[1] P. J. Werbos, ‘‘Backpropagation through time: What it does and how to do it,’’ Proc. IEEE, vol. 78, no. 10, pp. 1550–1560, Oct. 1990.

[2] D. Parker, ‘‘Learning Logic,’’ Massachusetts Inst. Technol., Cambridge, MA, USA, Tech. Rep. tr-47, 1985.

[3] Zhou, C., Wang, L., Zhang, Q. and Wei, X. (2013). Face recognition based on PCA image reconstruction and LDA. Semantic Scholar. [online] Available at: https://www.semanticscholar.org/paper/Face-recognition-based-on-PCA-image-reconstruction-Zhou-Wang/1799ad1b57130bbdcbfe9bb31e8157634c6369da.

[4] GeeksforGeeks.(2019).Python | Grayscaling of Images using OpenCV.[online]Available at: https://www.geeksforgeeks.org/python-grayscaling-of-images-using-opencv/ [Accessed 29 Apr. 2022].

[5] Shah, T. (2020). About Train, Validation and Test Sets in Machine Learning.[online] Medium. Available at: https://towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7#:~:text=The%20validation%20set%20is%20used%20to%20evaluate%20a [Accessed 29 Apr. 2022].M. Young, The Technical Writer’s Handbook. Mill Valley, CA: University Science, 1989.

[6] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. Data mining and knowledge discovery, (submitted), 1998.

[7] V. Vapnik. The nature of statistical learning theory. Springer, New York, 1995.

[8] Scikit-learn.org. (2019). sklearn.svm.SVC — scikit-learn 0.22 documentation. [online] Available at: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html.

[9] Team, K. (n.d.). Keras documentation: About Keras. [online] keras.io. Available at: https://keras.io/about/.