Computer Vision Report

Data

We created facial emotion detection algorithms by analysing images of a broad range of faces and classifying their emotions into seven categories/labels (1. surprise, 2. fear, 3. disgust, 4. happiness, 5. sadness, 6. anger, 7. neutral).

The collection includes 12,271 training and 3,068 test images, all of which have been scaled to 100*100. This is separated into two files, one of which will be used for the facial recognition video. We also clean the data file names as the images have 'Aligned' word which is removed.

Our initial hurdle was to combine the images with the labels which was completed in Python using the panda's package. These were provided on both the training and testing datasets and the test sets were utilised for comparison purposes.

There is an unequal number of expression categories in the dataset (Table 1) as the frequency in the table shows that the dataset is imbalanced across both train and test. We notice that Label 4. Happiness dominates and therefore there maybe bias and a higher skewness to this category, whereas the opposite is true for label 2. Fear. Having experimented with balancing the dataset using SMOTE, I found resampling saw a decrease in the overall accuracy and an increase in the minority classes, but the overall accuracy did not change significantly. Hence, I decided to proceed with the unbalanced data.

Feature descriptors are employed for our models and the extracted features are then classified; however, feature descriptors are not used for CNN models as they can automatically extract features. The model which provides the highest accuracy score is then applied and tested in the video section ("In the wild"). Our dataset on which we train our models is incompatible with the video format, hence face detection (Haar Cascade) methods are exploited. The data pipelines are as shown below.

Input Dataset → Feature Extractor → Classification Model → Results Output (FER Image Pipeline)
Input Video → Face Detection → FER → Results Output (FER Video Pipeline)

Implemented methods

We classify Two feature descriptors (HOGS & SIFT) and two classifiers (SVM & CNN) are used in this report.

Classifiers (SVM) - Support Vector Machines (SVM) is a supervised learning algorithm which was implemented as this model can be used for multi-classification problems as we have 7 emotion labels. It is a good classical classification model which usually gives good results when combined with HOG or SIFT. Each data item is plotted as a point in n-dimensional space, where n is the number of features, and each feature is the value of a specific co-ordinate. The classification is then carried out by locating the hyperplane that distinguishes between the classes. They can categorise new test data after giving an SVM model a set of labelled training data for each category. This is classified using the Scikit-Learn Library.

We first initialise the model in a random state by taking the model inputs and the images are then read into a single array and labels are loaded and read on both train and test data. Since SVM receives inputs of the same size, all images should be resized to a fixed size before being fed into the SVM. Key interest points (Using Hogs and Sift) and extract these descriptors from the facial images. We also changed the images to grayscale as doing this reduces computations required and simplified the algorithm. The model is constructed by creating a support vector classifier and then performing a 5-fold Cross Validation grid search. Model training involves using training data to train the model and testing data to test the model. We fit the estimator model on the training set, such that we find the best parameters that minimises the error and gives us the best accuracy on the unseen test dataset. The tuned parameters were Kernel {Linear, RBF}; C {0.01, 10}; Gamma {0.01, 0.001}.

Classifiers (CNN) - Convolution Neural Network is a type of deep neural network mainly used in the field of computer vision. IT is made up of two layers; Feature extraction/hidden layers consisting of convolutions and pooling and the classifier. The convolution is completed on the input data to generate a feature map with the support of a filter. After a convolution layer, the pooling layer is introduced, which decreases the number of parameters and computations, reducing training time and minimizing overfitting. Max-pooling is a technique that reduces the feature map size while retaining crucial information by taking the maximum value in each window. During training, a drop out is employed to ignore randomly selected neurons. This decreases the model's overfitting. We follow a similar method to read the images into a single array. We input variables for image width and height (100x100), batch size and epochs. We next configure our model through which the features are passed to train the model. We then test using the test features. Four convolution 2d layers and three Max Pooling Layers are used alongside dropout, flatten and dense layers. We used activation function of 'Relu' to filter negative values from the image and replaced with a zero. Softmax is used as it is a multi-class classification problem. We used a differing number of neurons in each layer with different kernel and pool size. To the compile the CNN model, the optimiser used is 'Adam', Learning Rate = 0.0001, Decay = 1e-6 and the metric to test is 'Accuracy'. The test set is used for the validation data. We then fit the model. This is classified using the Keras library.

Feature Extractors (SIFT) - Scale Invariant Feature Transform (SIFT) is a feature extraction approach that transforms image content into local feature coordinates that are highly distinctive, scale, and other image transformations. We use OpenCV for (SIFT) and this is combined with the SVM. SIFT was chosen because the features are resistant to occlusion and clutter, we can extract a large number of features from small objects, and individual features may be matched to a broad dataset of objects.

Feature Extractors (HOGS) – Histograms of Oriented Gradients is based on linear SVM. It measures the distribution of intensity gradients to describe the look and shape of an object in an image. The input image is divided into small intervals known as cells, and a histogram of gradient for the pixels within each cell is created. The value calculated from a larger region called a 'Block' is used to normalise all cells. The number of descriptors in this case is restricted as our input is set to 100x100 image. The number of orientation bins =8; Cell size = 16x16 and Block size= 1x1. Scikit Image is used here for HOGS.

'In the wild' Video Recognition – We apply our best performing model to our chosen video. Pre-processing must be complete to ensure the video is compatible with the model being applied. This is because the images in the dataset are 100x100 pixels. We follow the data pipeline as mentioned earlier and apply a facial detection algorithm (Haar cascade). Once the faces are detected, we do facial emotion recognition by using our best performing classification model. The results of the video show this with a box to detect the face and text to show the changes in emotions in the video. This is saved as a function EmotionRecognitionVideo in the Test_Functions.ipynb file.

Results

		Dataset Frequency	
Label	Label2	Train	Test
1	Surprise	1290	329
2	Fear	281	74
3	Disgust	717	160
4	Happiness	4772	1185
5	Sadness	1982	478
6	Anger	705	162
7	Neutral	2524	680

Table 1. Label Freq

Classifier	Feature	Hyperparameters (kernel;gamma;C)	Accuracy	Time (seconds)
SVM	SIFT	{'C': 10, 'gamma': 0.01, 'kernel': 'rbf'}	0.39	718
SVM	HOGS	{'C': 10, 'gamma': 0.01, 'kernel': 'rbf'}	0.62	31
CNN	-	{'Epoch': 30}	0.76	810

Table 2. Tuning and results

SVM+SIFT (SS) **SVM+HOGS (SH)** **CNN**

	precision	recall	f1-score	support
1	0.00	0.00	0.00	309
2	0.00	0.00	0.00	73
3	0.00	0.00	0.00	151
4	0.39	1.00	0.56	1125
5	0.00	0.00	0.00	453
6	0.00	0.00	0.00	156
7	0.00	0.00	0.00	606
accuracy			0.39	2873
macro avg	0.06	0.14	0.08	2873
weighted avg	0.15	0.39	0.22	2873

	precision	recall	f1-score	support
1	0.53	0.50	0.51	329
2	1.00	0.07	0.13	74
3	0.57	0.05	0.09	160
4	0.71	0.87	0.78	1185
5	0.53	0.43	0.47	478
6	0.62	0.33	0.43	162
7	0.54	0.65	0.59	680
accuracy			0.62	3068
macro avg	0.64	0.41	0.43	3068
weighted avg	0.62	0.62	0.59	3068

	precision	recall	f1-score	support
Surprise	0.78	0.77	0.78	329
Fear	0.73	0.43	0.54	74
Disgust	0.53	0.32	0.40	160
Happiness	0.89	0.88	0.88	1185
Sadness	0.67	0.66	0.66	478
Anger	0.60	0.65	0.62	162
Neutral	0.68	0.78	0.73	680
accuracy			0.76	3068
macro avg	0.70	0.64	0.66	3068
weighted avg	0.76	0.76	0.76	3068

Table 3. Classification Reports

SVM+SIFT **SVM+HOGS** **CNN**

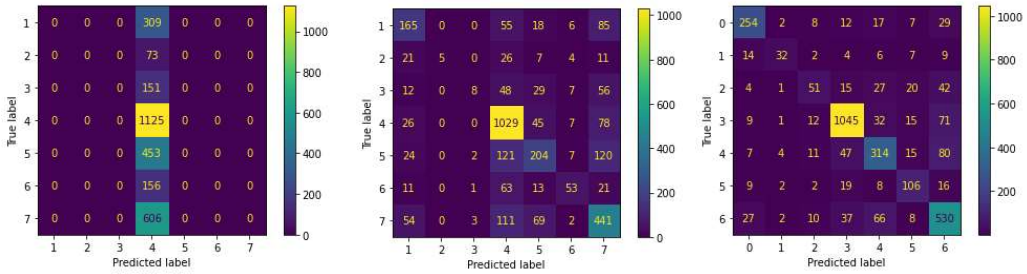
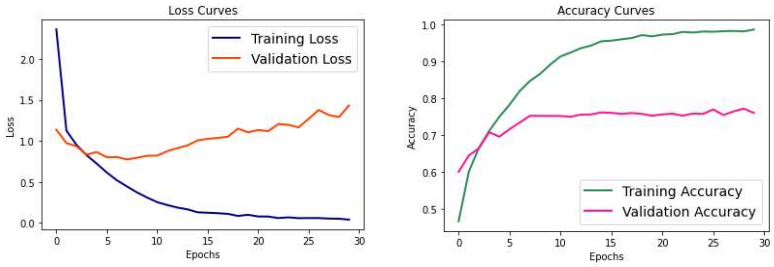


Figure 1. Confusion Matrices



Graphs 1. Accuracy and Loss Curves for CNN

SVM+SIFT **SVM+HOGS** **CNN**

Figure 2. Facial Emotion Label and Prediction Results

CNN Model applied to video

Figure 3. Screen Captures from the 'In the wild' Video

Discussion

Table 2. depicts the best hyperparameters required to achieve that resulted in the best accuracies for each model. We notice that the hyperparameters for SVM to achieve the best results are the same irrespective of which feature descriptor is used. SVM achieved its best performance on C as 10, Gamma as 0.01 and

kernel as rbf. The CNN model worked best when the epochs selected were 30. SVM and SIFT returned the lowest accuracy of 0.39 and 718 seconds for time taken to train the model. SVM and HOGS returned a higher accuracy of 0.62 with a much less time taken to execute at 31 seconds. CNN achieved an accuracy of 0.76 (Epochs 30) with time taken of 810 seconds. This shows that CNN is an algorithm that can provide high accuracy scores but at the expense of being computationally heavy. Changing Epochs to 15 reduced the accuracy very slightly to 0.75 but time taken significantly reduced to 118 seconds. CNN was faster as it utilised GPU on colab. We can conclude that the features selection in SVM makes a positive difference to the accuracy and time. The computation times for using gridsearch CV on hyperparameters for SVM and SIFT and SVM and HOGS were 12,661 and 13,913 respectively. Due to this limitation, a select few parameters were only applied. HOGs were the best feature descriptor with SVM and whilst it outperformed SVM with SIFT, it failed to outperform CNN.

Table 3. Shows us the classification reports for each model. SS achieved really low recalls across the emotion labels. As the data is imbalanced in each class of the train set (Table 1.), we notice low recalls for 'Fear', which are 0 for this model. Recall for happiness is a perfect 1, and as this has the most frequency, this gave rise to overfitting. The noise from training data is acquired to the degree at which it gives a negative impact on the new data. This is also confirmed by the confusion matrix (Figure 1.) that shows an overfitting to class 4: happiness, as these models tend to forecast oversampled classes. We see the recall scores better distributed for SH and CNN model although class 2: Fear is still pretty low for the SH model. The confusion matrixes show a better prediction to its true label as the frequency is distributed diagonally as required with CNN with the highest scores. These low recall scores that the model did not assign any emotions to any facial images.

As CNN provided our best results, we applied this model to the EmotionRecognitionVideo function. Figure 3. depicts the results of facial emotion recognition on video. We use a technique called Viola-Jones face detection or Haar cascades for face detection. This is an object detection algorithm (OpenCV), used to identify faces in an image or a real time video. We notice this face detection and CNN algorithm correctly applying the emotions of happiness and sadness to the images. I noticed that there were some emotions which were incorrectly recognised or not detected. Sometimes when multiple people are in the image, the model fails to recognise all emotions on multiple faces. This scene has a big change in mood, and this is captured by having recognising mainly happiness which turns to sadness. We determine that the model's performance on the video is sufficient. Figure 2. Shows the label and prediction on 4 random images from the test set. SS predicted 1/4 labels correctly, SH predicted 2/4 correctly and CNN correctly predicts 3/4 images. Graphs 1. Our model begins to overfit around 10 epochs, at which point our validation accuracy hits its maximum. The validation loss also starts to rise at this point. We can see that the val loss is 1.43 and the val accuracy is 0.7601 at Epoch 30 is reasonable. As a result, we run this model on the test data.

For future work, the models can be improved if the data is more balanced either using SMOTE or we can combine classes to narrow the labels. For example, Anger and Disgust are similar as well as Fear and Sadness emotions. These can be combined to get a more balanced distribution across the classes. We can improve the accuracy of SS model by using Bag of Visual Words or clustering using K-means which can help improve the accuracy, although earlier experiments did not show much difference in results. We can also apply a more extensive hyperparameter search across the models for any future work to improve accuracies. The Haar Cascade is a fast-computing face detector, but we could also experiment with different algorithms here such as MobileNet; ResNet as Haar Cascade is not the most accurate and is quite dated.