Facial Emotion Recognition on RAF-DB

Link to the the codes of this project: https://drive.google.com/drive/folders/119JmTTQVJNzoBkonu2gxchF2LFdd7A6d?usp=sharing

Chikaze Mori
City, University of London
Student ID: 200038013
Chikaze.Mori.2@city.ac.uk

***Abstract*—This paper applies different feature descriptors and classifiers and convolutional neural network in a facial emotion recognition task and compare the performances of the models. The model that achieves the best accuracy is to be applied to a video along with face detection and alignment methods.**

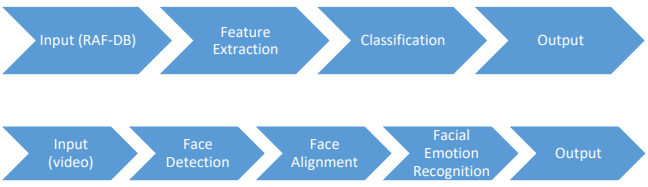
Keywords—facial emotion recognition, convolutional neural network

I. Introduction

In this paper, we develop several facial emotion recognition algorithms, which take facial images as inputs and classify their emotions. These algorithms are to be applied into the dataset given [1], which contains 12,271 and 3,068 aligned facial images labelled with one of the 7 different basic emotions for training and testing respectively, and the results on the test set are to be compared. The best performed model is to be tested on the video given [2], which is a movie clip.

First feature extraction is conducted on RAF-DB by applying a feature descriptor, and the extracted features are to be classified, however, feature descriptors are not used for any CNN models as they can extract features by themselves. After implementing all the models, we apply the best performed model into the video, however, face detection and alignment methods are to be applied to the video before our best model as the format of the video is not compatible with RAF-DB on which we train our models. These processes are illustrated in the figure below:

Figure 1: Pipelines of our methodology for facial emotion recognition on RAF-DB (top) and a video (bottom)



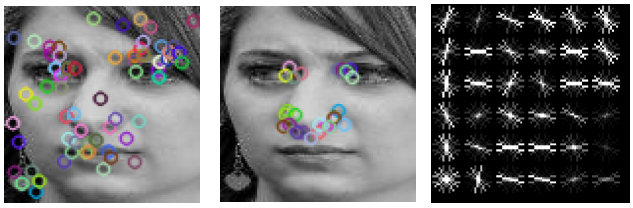
II. Theoretical Elements and Implementation Details

For comparison, we implement 3 feature descriptors and 3 classifiers for each feature descriptor, which provide us 9 different combinations. For each combination, we apply grid search with 5-fold cross validation, and only the best performed model is to be tested. For convolutional neural network (CNN) models, we implement a few residual neural network models with different number of epochs.

A. Feature Descriptors

Feature descriptors are algorithms that extract keypoints of images where information is highly concentrated. We apply 3 feature descriptors into the dataset, SIFT, ORB and HOG.

Figure 2: Examples of SIFT (left), ORB (middle) and HOG (right)



We use the open-source library, OpenCV [3] for SIFT and ORB, and the python image processing library, Scikit-image [4] for HOG.

1) Scale Invariant Feature Transform (SIFT)

SIFT is a method that transforms image data into scale-invariant coordinates relative to local features [5]. It produces a set of Difference of Gaussians (DOG) images that are the subtractions of adjacent Gaussians images with two different sigma, σ and $k\sigma$:

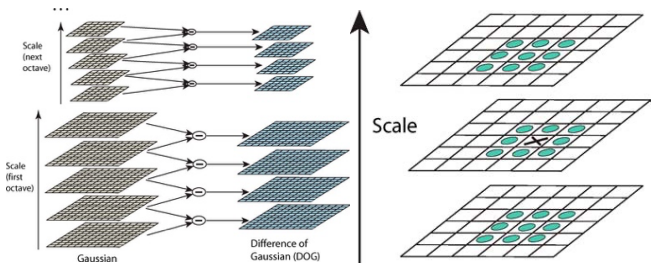
$$G(x, y, k\sigma) - G(x, y, \sigma) \approx (k - 1)\sigma^2\nabla^2 G,$$

where $\sigma^2\nabla^2 G$ is the scale-normalized Laplacian of Gaussian and k is a constant multiplicative factor. This process, shown in Fig. 3, is done for each octave and repeated for down-sampled images.

Once DOG images are found, the local maxima and minima of the images are detected by comparing each sample point to its 8 neighbours in the current image and 9 neighbours in the adjacent scales (Fig. 3).

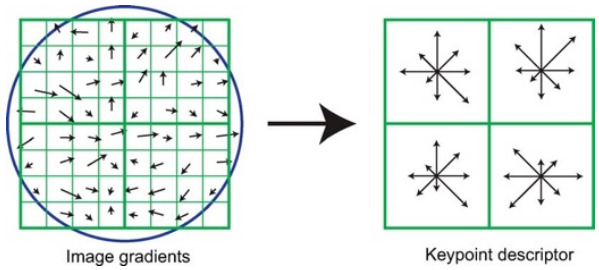
After those local points are found, detailed fits are to be performed in order to localise more accurate keypoints. Firstly, Taylor series expansion of scale space is applied to achieve more accurate location. Subsequently, edges are to be removed by using Harris corner detector in order to eliminate the high responses of DOG to edges.

Figure 3: An image of DOG (left) and an image of local extrema detection (right) [5]



To each keypoint obtained thorough the previous processes, a consistent orientation is assigned based on local image properties, and the keypoint descriptor can be represented relative to this orientation and therefore achieve invariance to image rotation [5].
A keypoint descriptor is created by computing the gradient magnitude and orientation at point in a 16×16 neighbourhood around the taken keypoint (Fig. 4).

Figure 4: An image of how keypoint descriptors are created [5]



In this work, SIFT was applied to the dataset and the keypoint descriptors produced were stored in a new dataset. Instead of using this newly created dataset for training as it is, we applied mini-batch k-means clustering to the part of the new dataset made from the train set and assigned those descriptors to k clusters to reduce computational costs, where the batch size is the number that is the total number of descriptors divided by 4 and applied a floor function and k is the number of labels found in the dataset times 10. A new test set was created by assigning the key descriptors based on the clusters observed in the process of k-means clustering.
When implementing SIFT, the parameters to be considered are the number of best features to retain, the number of octave layers, contrast threshold used to filter out weak features, edge threshold and the sigma of the Gaussian, and the values used in our work are 0, 3, 0.04, 10 and 1.6 respectively. The example of SIFT applied to the dataset is shown in Fig. 2. In the figure, the keypoints are concentrated in the eyes, the nose and the mouth, where pixel differences are large.

2) Oriented FAST and rotated BRIEF (ORB)

ORB is a very fast binary descriptor based on FAST keypoint detector [6] and BRIEF descriptor [7]. It was proposed to enhance many common image-matching applications, e.g., to reduce the computational time for feature-based object detection on standard PCs without GPU acceleration, and yet performs as well as SIFT [8]. It uses FAST [6], which is a corner detection method that uses a circle of 16 pixels to classify if a candidate point is a corner, in order to find keypoints, and then filter these keypoints by Harris corner detector as FAST does not produce multi-scale features. For orientation, it computes the weighted intensity [9] centroid of the patch with located corner at centre, where the moment of a patch is defined as:

$$m_{pq} = \sum_{x,y} x^p y^q I(x, y) \ .$$

For a feature extraction, ORB uses BRIEF, which is an efficient feature descriptor that uses binary strings. BRIEF is very fast both to build and to match and outperforms SURF and U-SURF [7].
In this work, ORB was used to create a new dataset that contains key descriptors in the same way as SIFT was applied along with mini-batch k-means clustering. When implementing ORB, the parameters considered were the maximum number of features to retain, edge threshold and the size of the patch used for BRIEF, and the values used in our work were 500, 31 and 31 respectively. The example of ORB applied to the dataset is shown in Fig. 2. In the figure, keypoints are mainly around the inner corner of the eyes and the nostrils, unlike SIFT whose keypoints were more scattered.

3) Histograms of Oriented Gradients (HOG)

HOG is a linear SVM based feature descriptor that significantly outperforms existing feature sets for human detection [10]. It describes local object appearance and shape of an image by measuring the distribution of intensity gradients. The input image is divided into smaller regions called cells and a histogram of gradient directions is created for the pixels within each cell. All cells are to be normalised by the value calculated from the intensity across a larger region called a block.
In this work, HOG was applied to the dataset and a new dataset that contains the key descriptors was created. As our input is set to be a 100×100 pixels image, the number of descriptors made through HOG is also limited, thus, it is not necessary to implement k-means clustering over the descriptors unlike SIFT and ORB. The parameters used for our HOG implementation were 8, 16×16 and 1×1 for the number of orientation bins, the size of a cell and the size of a block respectively. The example of HOG applied to the dataset is shown in Fig. 2. The figure vaguely looks like a face with a mouth distinctively shaped.

B. Classifiers

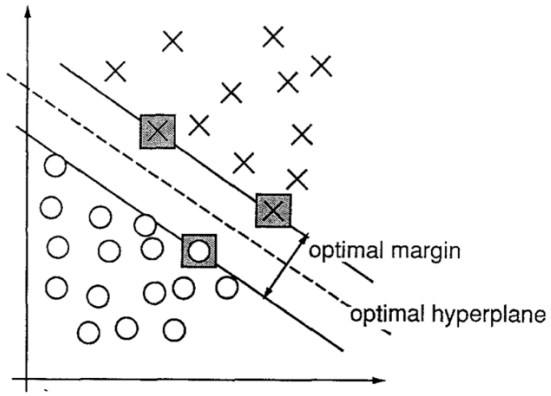
Classifiers are algorithms used to predict the class of given data. In our work, classifiers are to be applied into the datasets created by the feature descriptors. In other words, we classify facial emotion of the images by classifying the extracted features of them. All the classifiers are applied by using the python machine learning library, Scikit-learn [11].

1) Support Vector Machines (SVM)

SVM is a supervised learning model and one of the classical machine learning techniques that can still help solve big data classification problems, which was first developed and

introduced by Vapnik in 1995 [12]. Conceptionally, the implement is that input vectors are non-linearly mapped to a very high-dimension feature space where a linear decision surface is constructed [9]. For multi-class classification tasks, SVM can use either One-vs-One or One-vs-Rest strategy. SVM assigns samples to one of two categories and maximise the width of the gap between the two categories (Fig. 5).

Figure 5: An example of a separatable problem in a 2-dimensional space [12]

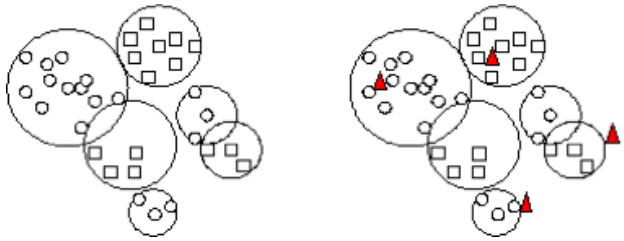


It can help the multidomain applications in a big data environment, however, SVM is mathematically complex, and it is computationally expensive to reach the SVM solution by stochastic gradient descent, mainly due to the regularisation term [12].
In this work, SVM was applied to the datasets created by SIFT, ORB and HOG. Grid search with 5-fold cross validation was conducted for each model, and the hyperparameters tuned were the kernel type and the regularisation parameter C, with {Linear, RBF} and {1, 10} respectively. The best model among all the combination of grids for each model is to be tested and we compare the results.

2) K-Nearest Neighbours (KNN)

KNN is a classifier first developed by Fix and Hodges in 1951 [13], which can be used for both classification and regression. It is a non-parametric method that is simple but effective [14]. First, KNN calculates the similarity for each data to all representatives in the model. Based on the similarity and Euclidean Distance, each data is to be classified to a category (Fig. 6).

Figure 6: Examples of KNN [14]

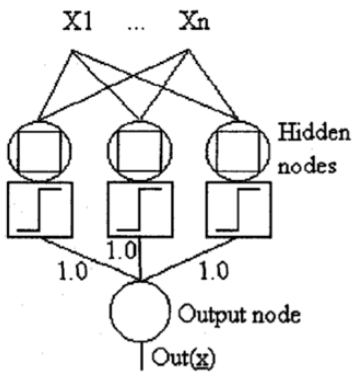


In this work, KNN was applied to the datasets created by SIFT, ORB and HOG. Grid search with 5-fold cross validation was conducted for each model, and the hyperparameters tuned were the weight function used in prediction and the leaf size, with {Uniform, Distance} and {15, 30} respectively. For every model, the number of neighbours was set to be 7, corresponding to the number of the classes. The best model among all the combination of grids for each model is to be tested and we compare the results.

3) Multi-Layer Perceptron (MLP)

MLP is a type of artificial neural network that consists of multiple fully connected layers of nodes, an input layer, one or more hidden layers and an output layer (Fig. 7). Hidden layers are to apply weights to the inputs and direct them through an activation function as the output. Based on the amount of error in the output compared to the expected result, the weights in the perceptron are to be modified thorough the process of backpropagation [15].

Figure 7: An example of MLP with 2 layers [15]



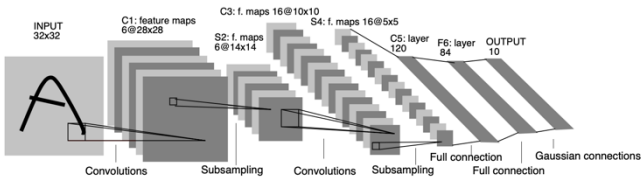
MLP is proven to be a universal approximator [16] that does not make any assumption regarding the underlying probability density functions or other probabilistic information about the pattern classes under consideration in comparison to other probability-based models [15].
In this work, MLP was applied to the datasets created by SIFT, ORB and HOG. Grid search with 5-fold cross validation was conducted for each model, and the hyperparameters tuned were the regularisation term parameter and the learning rate, with {0.0001, 0.001} for both. For every model, maximum number of iterations was set to be 200. The best model among all the combination of grids for each model is to be tested and we compare the results.

C. Convolutional Neural Network (CNN)

CNN is a type of deep neural network mainly used in the field of Computer Vision, which can take images as inputs, assign importance to various objects of the images and differentiate them. CNN is a combination of 3 architectural ideas to ensure some degree of shift, scale, and distortion invariance: local receptive fields, shared weights and spatial sub-sampling

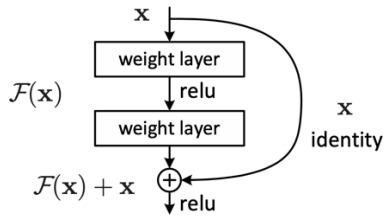
[17]. A typical architecture of CNN is shown in the figure below:

Figure 8:An example of CNN (LeNet-5) [17]



The model of CNN used in our work is residual neural network, which is for short, ResNet. It has significantly deeper network, and yet does not cause degradation problem utilising skip connections, or shortcuts to jump over some layers (Fig. 9).

Figure 9: An image of residual learning [18]



In this work, we use the ResNet18 model provided by the open-source python machine learning library, PyTorch [19]. It consists of 18 layers, including a max pooling layer, 16 convolutional layers and an average pooling layer (Tab. 1). We train 2 ResNet18 models for different number of epochs, 5 and 20, and only the better performed model is to be tested and compared with the other models.

Table 1: Examples of architectures of ResNet [18]

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
		3×3 max pool, stride 2				
conv2_x	56×56	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

D. Facial Emotion Recognition on the Video

Once the best model is chosen, we apply it to the video. To do so, it is necessary to process the video and make it compatible to the model. As the images of the dataset are faces aligned according to the location of the eyes and the centre of the mouth and resized to 100×100 pixels, we conduct similar processes into the video frames (Fig. 1). First the video is divided into single frames, and to each frame, we apply the facial detection algorithm provided by OpenCV [3]. Once faces are detected, two eyes in the face are to be detected using the eye detector also provided by OpenCV [3]. According to those eyes detected, the faces are aligned using similarity transformation. The centre of mouth is not used for this alignment for computational convenience. Faces are not aligned at all when two eyes cannot be detected. After these procedures, faces are to be resized to 100×100 pixels and put into a new dataset. To this new dataset that contain faces, the best model chosen is applied and the faces are to be classified as one of the emotions. Those results of classification are to be displayed in the video as a text along with a rectangle surrounding the face. This process is stored in the function EmotionRecognitionVideo, which is available on the link shown at the top of this paper.

E. Failed Approach

We also attempted to train a ResNet50 model for 20 epochs, however, we did not succeed as it was computationally too expensive and ResNet50 seemed to require more epochs than 20, in which ResNet18 could achieve a high accuracy. Applying CNN models pretrained on ImageNet was also our option, however, we did not proceed as the time for this project was limited. We constructed our own CNN model, however, we did not train it to save time.

III. Discussion of the Obtained Results

Tab. 2 shows the hyperparameters that achieved the best accuracy for each model. As shown, the best hyperparameters were the same for each classifier regardless which feature descriptor they used. For example, SVM achieved the best performance, when the kernel type was RBF and the regularisation parameter C was 1 with any feature descriptor. Among the 2 CNN models, the one trained for 5 epochs achieved higher accuracy than the one trained for 20 epochs.

This suggests that the feature descriptors did not significantly change the data structure itself and influence classifiers, which is consistent with our methodology as the feature descriptors were only to extract features and did not purposefully change the number of datapoints, on which hyperparameters such as learning rate is highly dependent.

As the CNN model trained for 20 epochs could not overperform the one trained for only 5 epochs, we consider the possibility that the model might have overfitted, considering the fact that the train set is imbalanced as shown in Tab. 3. For example, the train set has only 281 images labelled as fear while it has 4772 images of happiness.

Table 2: Best hyperparameters for each model

Feature Descriptor	Classifier	Hyperparameters
SIFT	SVM	RBF, 1
SIFT	KNN	30, uniform
SIFT	MLP	0.0001, 0.001
ORB	SVM	RBF, 1
ORB	KNN	30, uniform
ORB	MLP	0.0001, 0.001
HOG	SVM	RBF, 1
HOG	KNN	30, uniform
HOG	MLP	0.0001, 0.001
	CNN	5

Table 3: Number of each emotion in the train set

Emotion	Number
Surprise	1290
Fear	281
Disgust	717
Happiness	4772
Sadness	1982
Anger	705
Neutral	2524

Fig. 11 is the two sample images from the test set which are faces labelled with surprise and anger, and Tab. 4 shows the predictions that each model made for those sample images. As shown in Fig. 11, those two faces seem quite neutral despite the emotions labelled and classifying those correctly is arguably difficult. In Tab. 3, the emotions correctly predicted are in red.

In fact, most models predicted the emotions of those faces wrong, and only the CNN model successfully classified both correct. For the left image labelled as surprise, all the HOG models predicted as neutral and the other models predicted as happiness, besides the KNN models of SIFT and ORB correctly classifying it. For the right image labelled as anger, most models predicted as happiness even though the face does not look happy at all. These suggest that all the models, except for the CNN model, was highly affected by the imbalances of the train set that has overwhelmingly more images labelled as happiness and neutral.

Figure 10: Examples of images from the test set (left: surprise, right: anger)

Table 4: Predictions for both sample images made by each model

Feature Descriptor	Classifier	Left image	Right image
SIFT	SVM	Happiness	Happiness
SIFT	KNN	Surprise	Happiness
SIFT	MLP	Happiness	Happiness
ORB	SVM	Happiness	Happiness
ORB	KNN	Surprise	Surprise
ORB	MLP	Happiness	Happiness
HOG	SVM	Neutral	Neutral
HOG	KNN	Neutral	Anger
HOG	MLP	Neutral	Neutral
	CNN	Surprise	Anger

Fig. 12 is the output of EmotionRecognition function for HOG-SVM model, which is available on the link shown at the top of this paper. The function takes a path to the test set and a name of a model as inputs and shows 4 randomly picked images from the test set with the correct labels and the predictions of the model for the image. Samples for the other models can be displayed by using the function.

Figure 11: Examples of facial emotion recognition by the HOG-SVM model

Tab. 4 shows the time consumed for training and the accuracy achieved with the best hyperparameters for each model, where the time column represents how long it took to train the model adding to the time spent for extracting the features.

As shown in red in the figure, the highest accuracy among all the models, 74%, was achieved by the CNN model. Compared to the others, this accuracy is astonishing, however, the time consumed for training the model is also astonishingly high, 8557.9 seconds, which is more than 2 hours. This suggests that CNN is a very powerful algorithm that requires a lot of computation.

HOG was the best feature descriptor with any classifier, which was predictable as its strength is in human detection [10], yet it could not overperform the CNN model. ORB is the fastest feature descriptor while those 3 models used ORB achieved one of the lowest accuracies. This is consistent with the theories in terms of speed as ORB is designed to be fast [8], however, it did not perform as well as SIFT. It is suspected that the disappointing performance of ORB might have been caused by the thresholds for edges being too high and the size of the patch used for BRIEF being too large. SIFT was as slow as HOG compared to ORB, while achieving lower accuracies than HOG despite of being applied k-means clustering to improve efficiency. As we did not explore any other values for the hyperparameters of those feature descriptors, optimising those values would improve its accuracies and is recommended as a further study.

The best classifier varied depending on the feature descriptor, while KNN constantly achieved the lowest accuracies with any feature descriptor as its most important hyperparameter, the number of clusters, was not tuned. However, being trained for a relatively longer time, SVM achieved higher accuracies

than the others in overall, and the highest accuracy, 65%, with HOG. This suggests that SVM is a computationally heavy algorithm and yet is powerful even in the domain of image classification. Although it could not compete with CNN, SVM achieved the noticeably high accuracy with HOG with the training time more than 27 times shorter than CNN.

Table 5: Time spent for training and accuracy on the test set for each model

Feature Descriptor	Classifier	Time (second)	Accuracy
SIFT	SVM	101.0 + 237.7	0.42
SIFT	KNN	101.0 + 9.8	0.36
SIFT	MLP	101.0 + 130.6	0.42
ORB	SVM	41.3 + 219.0	0.39
ORB	KNN	41.3 + 9.7	0.28
ORB	MLP	41.3 + 133.6	0.38
HOG	SVM	102.3 + 441.4	0.65
HOG	KNN	102.3 + 12.2	0.53
HOG	MLP	102.3 + 208.2	0.62
	CNN	8557.9	0.74

Tab.6 shows precision, recall and F1 score of each model respectively. Because of the imbalances of each class in the train set, all the models except for CNN have the extremely low recalls for fear, which are mostly lower than 0.10. However, the F1 scores of the HOG models are relatively better for fear, and so are for disgust and anger, which are also terribly represented in the train set. The only exception, CNN, has the surprisingly high recall, 0.73, which is even higher than the other recalls of the model. This suggests that CNN and HOG were successful at distinctively extracting features of classes underrepresented, making them the best models among all.

Table 6: Precision, Recall and F1 score of each model
For fear, disgust and anger, which have small number of samples in the train set, the recalls of many of the models are 0, which suggest that those models did not classify any face as those emotions. In contrast, those models tend to predict the oversampled classes. For example, the recall of the ORB-SVM model for happiness is 0.96 while the recalls for surprise, fear, disgust, and anger are lower or equal to 0.03. In other words, the model did not predict at all but happiness.

Feature Descriptor	Classifier	Surprise	Fear	Disgust	Happiness	Sadness	Anger	Neutral
SIFT	SVM	0.45, 0.09, 0.15	0.00, 0.00, 0.00	0.00, 0.00, 0.00	0.43, 0.86, 0.57	0.28, 0.02, 0.04	0.00, 0.00, 0.00	0.35, 0.29, 0.32
SIFT	KNN	0.23, 0.21, 0.22	0.04, 0.03, 0.03	0.07, 0.03, 0.04	0.43, 0.75, 0.54	0.25, 0.13, 0.17	0.09, 0.07, 0.08	0.28, 0.08, 0.12
SIFT	MLP	0.30, 0.18, 0.22	0.00, 0.00, 0.00	0.11, 0.01, 0.01	0.46, 0.81, 0.59	0.32, 0.14, 0.19	0.29, 0.04, 0.08	0.37, 0.30, 0.33
ORB	SVM	0.22, 0.01, 0.01	0.00, 0.00, 0.00	0.00, 0.00, 0.00	0.39, 0.96, 0.55	0.30, 0.06, 0.10	0.00, 0.00, 0.00	0.29, 0.03, 0.05
ORB	KNN	0.15, 0.20, 0.17	0.06, 0.05, 0.06	0.09, 0.05, 0.07	0.45, 0.46, 0.45	0.19, 0.22, 0.20	0.11, 0.22, 0.14	0.27, 0.14, 0.19
ORB	MLP	0.33, 0.01, 0.02	0.00, 0.00, 0.00	0.00, 0.00, 0.00	0.39, 0.93, 0.55	0.22, 0.06, 0.10	0.00, 0.00, 0.00	0.31, 0.06, 0.10
HOG	SVM	0.59, 0.50, 0.54	1.00, 0.15, 0.26	0.64, 0.06, 0.10	0.73, 0.89, 0.80	0.56, 0.45, 0.50	0.64, 0.33, 0.44	0.57, 0.72, 0.64
HOG	KNN	0.47, 0.30, 0.36	0.80, 0.16, 0.27	0.13, 0.04, 0.07	0.63, 0.80, 0.70	0.40, 0.33, 0.36	0.36, 0.19, 0.25	0.47, 0.54, 0.50
HOG	MLP	0.52, 0.51, 0.51	0.83, 0.07, 0.13	0.42, 0.07, 0.12	0.68, 0.89, 0.77	0.50, 0.42, 0.46	0.50, 0.36, 0.42	0.57, 0.54, 0.56
	CNN	0.66, 0.83, 0.73	0.26, 0.73, 0.38	0.25, 0.46, 0.32	0.90, 0.86, 0.88	0.57, 0.68, 0.62	0.51, 0.69, 0.59	0.86, 0.63, 0.72

Overall, the CNN model was unbeatable in any perspective, except it takes significantly longer time compared to the others. That is to say, we conclude that CNN is the best model among those 10 different models, and we apply the trained CNN model to the given video, using EmotionRecognitionVideo function that is available from the link shown at the top of this paper.

Fig. 13 is the sampled images showing the outputs of facial emotion recognition on the video. For the top left image, where the character seems neutral, the model successfully detects the face and classify its emotion. Moreover, for the top right image, where the character is shown very small behind the blinds, the model still succeeds to detect and classify the face. However, for the bottom left image, where the character is covering his face with his hand, the model classifies it as disgust, even though the correct emotion is likely to be neutral. For the bottom right image, where multiple faces appear at the same time, the model detects more than half of the faces leaving the one shown the largest yet sided, however, the classifications for the faces are sceptical.

Figure 13: Examples of facial emotion recognition on the video

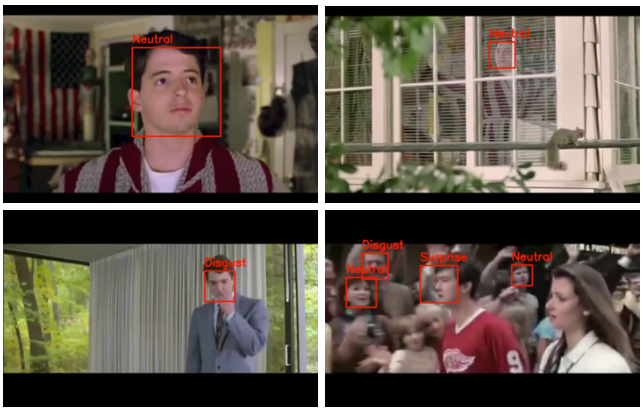
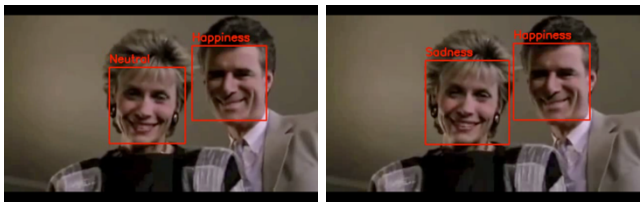


Fig. 14 is the other examples of facial emotion recognition applied to the video by using the best CNN model. The left image is one frame before the right image and both images contain the two faces of the same characters. Both characters seem happy in both images, however, the model fails to classify the faces of the woman on the left of the images and recognise them as neutral on the left image and sadness on the right image. As CNN is a deep neural network algorithm, we cannot know why the model predicted those happy faces as neutral or even sad, we conclude the performance of the model on the video is satisfactory.

Figure 14: Other examples of facial emotion recognition



IV. Conclusion

In this paper, we applied 10 facial emotion recognition algorithms into RAF-DB, compared the results and learned that CNN significantly outperforms the other models while requiring a lot of computation.

As time for this project was limited, we only trained 2 ResNet18 models, however, it is plausible to apply other popular CNN models such as AlexNet. In addition, with better GPU, training deeper CNN models such as ResNet50 or ResNet152 would be also plausible and expected to achieve higher accuracies when trained for enough number of epochs. Also, it is possible to improve accuracies simply using CNN models pretrained on ImageNet [20] provided by PyTorch [19].

To solve the problems caused by the imbalances of the classes in the train set, it would be practical to apply techniques such as under sampling, over sampling and SMOTE [21]. Besides these techniques, we suggest self-supervised learning methods such as SimCLR, which is a contrastive self-supervised learning algorithm that does not require specialised architectures or a memory bank [22]. By applying SimCLR, it is possible that we can boost up the learnings of the model on the underrepresented classes and achieve accuracies on those classes as high as on the classes often appeared in the train set.

References

[1] S. Li, W. Deng, and J. Du, 'Reliable Crowdsourcing and Deep Locality-Preserving Learning for Expression Recognition in the Wild', in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, Jul. 2017, pp. 2584–2593.

[2] 'Ferris and Friends on Vimeo'. https://vimeo.com/98084989 (accessed Apr. 28, 2021).

[3] Bradski G. The OpenCV Library. Dr Dobb's Journal of Software Tools. 2000.

[4] 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 Python. PeerJ 2:e453, 2014.

[5] D. G. Lowe, 'Distinctive Image Features from Scale-Invariant Keypoints', *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, Nov. 2004.

[6] E. Rosten and T. Drummond, 'Machine Learning for High-Speed Corner Detection', in *Computer Vision – ECCV 2006*, vol. 3951, A. Leonardis, H. Bischof, and A. Pinz, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 430–443.

[7] D. Hutchison *et al.*, 'BRIEF: Binary Robust Independent Elementary Features', in *Computer Vision – ECCV 2010*, vol. 6314, K. Daniilidis, P. Maragos, and N. Paragios, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 778–792.

[8] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, 'ORB: An efficient alternative to SIFT or SURF', in *2011 International Conference on Computer Vision*, Barcelona, Spain, Nov. 2011, pp. 2564–2571.

[9] P. L. Rosin, 'Measuring Corner Properties', *Computer Vision and Image Understanding*, vol. 73, no. 2, pp. 291–307, Feb. 1999.

[10] N. Dalal and B. Triggs, 'Histograms of oriented gradients for human detection', in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, Jun. 2005, vol. 1, pp. 886–893 vol. 1.

[11] Scikit-learn: Machine Learning in Python, Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011.

[12] C. Cortes and V. Vapnik, 'Support-vector networks', *Mach Learn*, vol. 20, no. 3, pp. 273–297, Sep. 1995.

[13] Fix, Evelyn; Hodges, Joseph L. Discriminatory Analysis. Nonparametric Discrimination: Consistency Properties. USAF School of Aviation Medicine, Randolph Field, Texas, 1951.

[14] G. Guo, H. Wang, D. Bell, Y. Bi, and K. Greer, 'KNN Model-Based Approach in Classification', in *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE*, vol. 2888, R. Meersman, Z. Tari, and D. C. Schmidt, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 986–996.

[15] Mu-Chun Su, Woung-Fei Jean, and Hsiao-Te Chang, 'A static hand gesture recognition system using a composite neural network', in *Proceedings of IEEE 5th International Fuzzy Systems*, Sep. 1996, vol. 2, pp. 786–792 vol.2.

[16] K. Hornik, M. Stinchcombe, and H. White, 'Multilayer feedforward networks are universal approximators', *Neural Networks*, vol. 2, no. 5, pp. 359–366, Jan. 1989.

[17] Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio, 'Object Recognition with Gradient-Based Learning', 1999.

[18] K. He, X. Zhang, S. Ren, and J. Sun, 'Deep Residual Learning for Image Recognition', *arXiv:1512.03385 [cs]*, Dec. 2015.

[19] Paszke, A. et al., PyTorch: An Imperative Style, High-Performance Deep Learning Library. In Advances in Neural Information Processing Systems 32. Curran Associates, Inc., pp. 8024–8035, 2019.

[20] J. Deng, W. Dong, R. Socher, L. Li, Kai Li and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," 2009 IEEE Conference on Computer Vision and Pattern Recognition, 2009.

[21] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, 'SMOTE: Synthetic Minority Over-sampling Technique', *jair*, vol. 16, pp. 321–357, Jun. 2002.

[22] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, 'A Simple Framework for Contrastive Learning of Visual Representations', *arXiv:2002.05709,* Jun. 2020.

APPENDIX

The larger size of some figures are shown below:

Figure 13: Examples of facial emotion recognition on the video

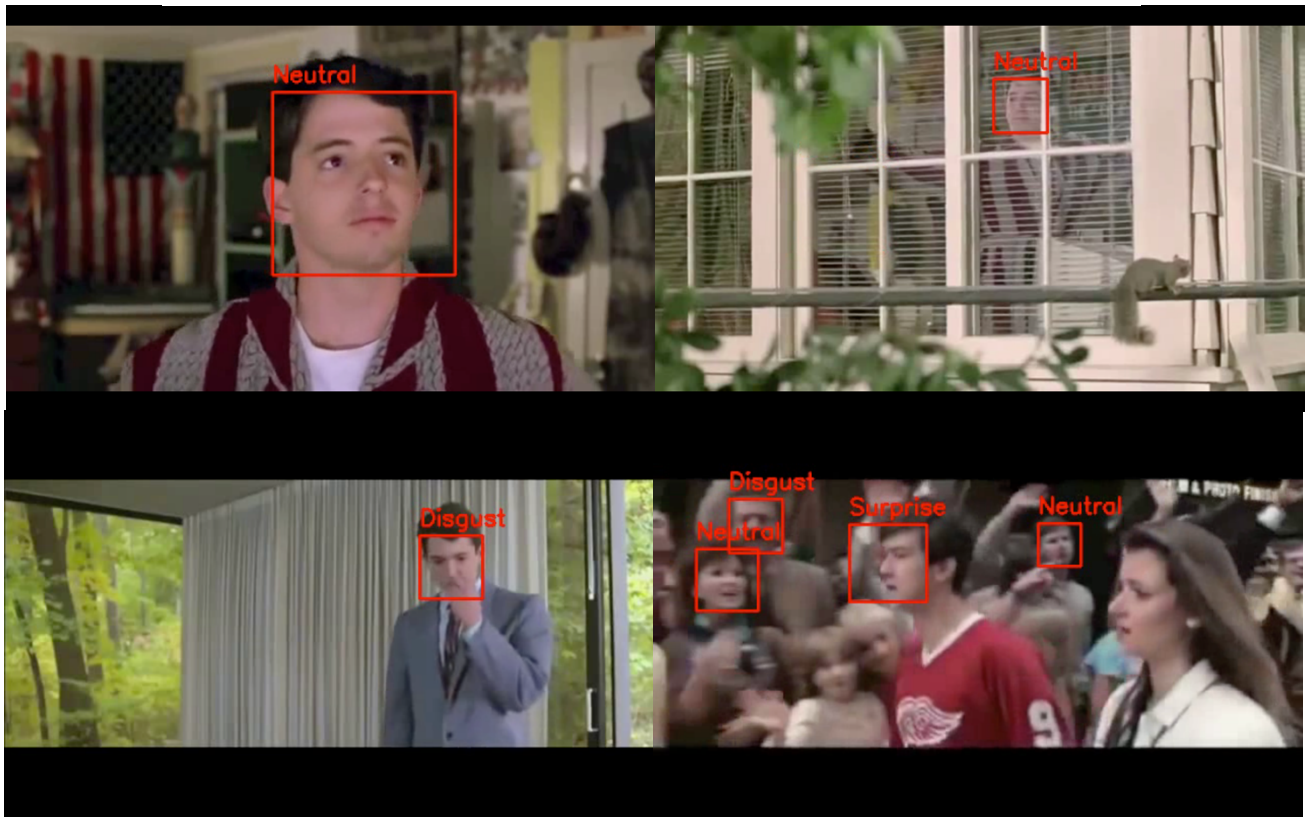


Figure 14: Other examples of facial emotion recognition

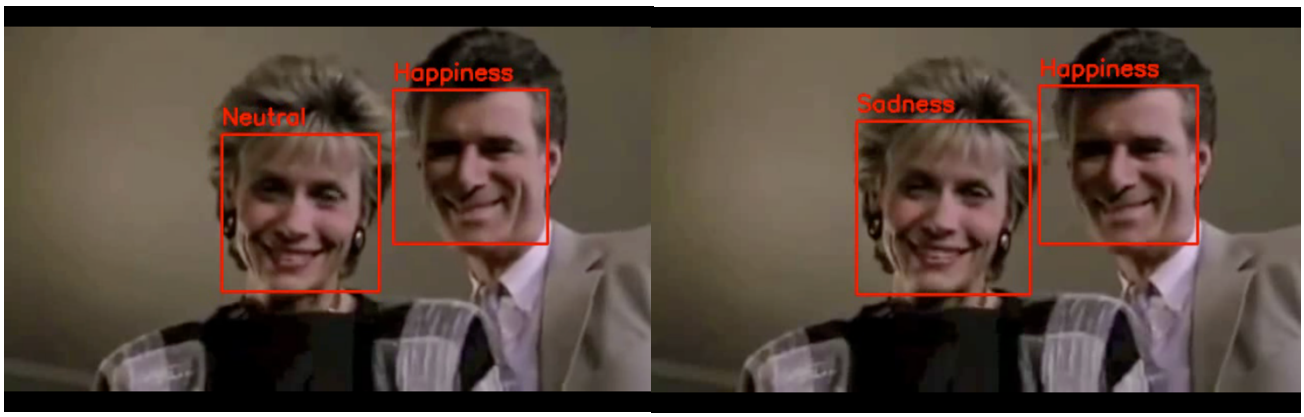


Table 6: Precision, Recall and F1 score of each model
For fear, disgust and anger, which have small number of samples in the train set, the recalls of many of the models are 0, which suggest that those models did not classify any face as those emotions. In contrast, those models tend to predict the oversampled classes. For example, the recall of the ORB-SVM model for happiness is 0.96 while the recalls for surprise, fear, disgust, and anger are lower or equal to 0.03. In other words, the model did not predict at all but happiness.

Feature Descriptor	Classifier	Surprise	Fear	Disgust	Happiness	Sadness	Anger	Neutral
SIFT	SVM	0.45, 0.09, 0.15	0.00, 0.00, 0.00	0.00, 0.00, 0.00	0.43, 0.86, 0.57	0.28, 0.02, 0.04	0.00, 0.00, 0.00	0.35, 0.29, 0.32
SIFT	KNN	0.23, 0.21, 0.22	0.04, 0.03, 0.03	0.07, 0.03, 0.04	0.43, 0.75, 0.54	0.25, 0.13, 0.17	0.09, 0.07, 0.08	0.28, 0.08, 0.12
SIFT	MLP	0.30, 0.18, 0.22	0.00, 0.00, 0.00	0.11, 0.01, 0.01	0.46, 0.81, 0.59	0.32, 0.14, 0.19	0.29, 0.04, 0.08	0.37, 0.30, 0.33
ORB	SVM	0.22, 0.01, 0.01	0.00, 0.00, 0.00	0.00, 0.00, 0.00	0.39, 0.96, 0.55	0.30, 0.06, 0.10	0.00, 0.00, 0.00	0.29, 0.03, 0.05
ORB	KNN	0.15, 0.20, 0.17	0.06, 0.05, 0.06	0.09, 0.05, 0.07	0.45, 0.46, 0.45	0.19, 0.22, 0.20	0.11, 0.22, 0.14	0.27, 0.14, 0.19
ORB	MLP	0.33, 0.01, 0.02	0.00, 0.00, 0.00	0.00, 0.00, 0.00	0.39, 0.93, 0.55	0.22, 0.06, 0.10	0.00, 0.00, 0.00	0.31, 0.06, 0.10
HOG	SVM	0.59, 0.50, 0.54	1.00, 0.15, 0.26	0.64, 0.06, 0.10	0.73, 0.89, 0.80	0.56, 0.45, 0.50	0.64, 0.33, 0.44	0.57, 0.72, 0.64
HOG	KNN	0.47, 0.30, 0.36	0.80, 0.16, 0.27	0.13, 0.04, 0.07	0.63, 0.80, 0.70	0.40, 0.33, 0.36	0.36, 0.19, 0.25	0.47, 0.54, 0.50
HOG	MLP	0.52, 0.51, 0.51	0.83, 0.07, 0.13	0.42, 0.07, 0.12	0.68, 0.89, 0.77	0.50, 0.42, 0.46	0.50, 0.36, 0.42	0.57, 0.54, 0.56
	CNN	0.66, 0.83, 0.73	0.26, 0.73, 0.38	0.25, 0.46, 0.32	0.90, 0.86, 0.88	0.57, 0.68, 0.62	0.51, 0.69, 0.59	0.86, 0.63, 0.72