# Facial Expressions Detection

Basant Abdelaal
Computer Engineering Department
The American University in Cairo
basantelhussein@aucegypt.edu

Nada Ayman Atia
Computer Engineering Department
The American University in Cairo
daliawk@aucegypt.edu

Youssef Hussein
Computer Engineering Department
The American University in Cairo
youssefhussein@aucegypt.edu

***Keywords—CNN, YOLO, HOG, SVM, Random Forrest, 68-PIE-landmarking scheme***

## I. Introduction

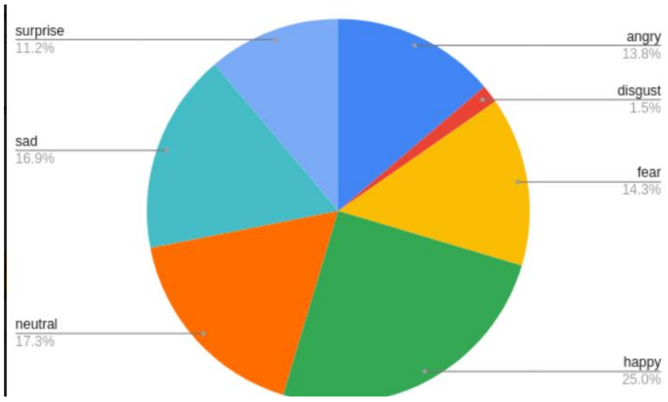
Emotions play an important role in our life. Emotions help us exhibit our feelings as not all of our thoughts can be conveyed through speech. By looking at the emotions of a person, we can decide the behavior of that person. Emotional reactions differ from person to person; no two people will experience the same emotional state in the same scenario.

Facial expressions have been studied since ancient times, making them a recognized and vital nonverbal communication channel. Furthermore, the evidence suggests that when conducting emotion recognition, facial expressions are frequently integrated with other modalities. Darwin's work on the universality of emotional facial expressions across nations and tribes laid the groundwork for the empirical study of facial expressions[1]. As a result, it made facial expressions the sole measure with developed frameworks, as they have been extensively explored in recent decades.

In the 1970's, research work has been done for recognizing the facial expression and they defined six basic categories of facial expressions: sadness ,happiness, fear, surprise, disgust and anger which has helped for the research of current expression recognition. Antient techniques depended only on traditional Computer vision techniques. In this paper, we used different approaches to solve the problem using classification and detection.

## II. Data Sets

The problem of facial expressions detection has many large datasets with hundreds of thousands of labeled images. However, such clean datasets are only available to the research community with no access for students. With that said, in this work, a publicly available dataset was used. The dataset is from Kaggle, and contains 7 classes: angry, disgust, fear, happy, neutral, sad, and surprise.
The dataset contains 35887 images. It is splitted to training data of size 28821 images, and validation data of size 7066. However, we resplitted the whole data to be 15% validation and 85% training.
Although the problem is purely related to faces, there was some challenge in this dataset as it contained cartoon images, images with no faces at all (like candles to represent sadness for example). However, those were a minor number of images compared to the whole dataset.
Another challenge was to label the data with bounding boxes for YOLO detection. This was automatically as explained in the approach section.
Below is a pie chart representing the numbers of images for each class.

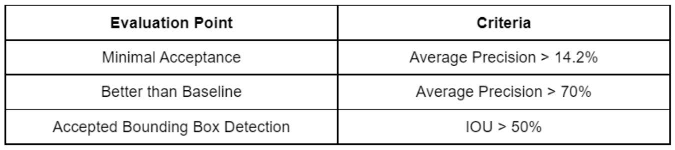


## III. Evaluation Criteria

We defined two problems in this work, classification task and detection (with bounding box) task. Our aim in this work is to produce good precision results in both tasks. Also, having good IOU (intersection over area) and mAP (mean Average Precision) for the detection task.
Since we have 7 classes, we expect the classifiers to exceed the accuracy of being random, which is basically 100/7 = 14.2%. This is the minimal criteria to be acceptable.

Due to the aforementioned challenges in the dataset, one cannot compare this work to the baseline in the "facial expression detection" problem in academia. Instead, we selected our baseline to be the best model implemented to solve this problem using this dataset in Kaggle. This is a proper way to compare our results since they used the same dataset.

The highest average precision that was found on Kaggle was 70% for Ayush Raj. And this defines the baseline for this work.
Moreover, for the detection task, there were no works to be compared to. Therefore, we defined it to have at least more than 50% IOU.
The table below summarizes our criteria:



| Evaluation Point | Criteria |
|---|---|
| Minimal Acceptance | Average Precision > 14.2% |
| Better than Baseline | Average Precision > 70% |
| Accepted Bounding Box Detection | IOU > 50% |

## IV. Approach

After extensive reviewing for the literature, it was decided to investigate 3 different approaches to tackle this problem. First approach is basically combining computer vision flavor with machine learning classifier. Therefore, it contains feature generation and image processing. Second approach is involved with neural networks, namely CNN as it has always yielded prosperous results in computer vision problems. Third approach was experimenting with the state-of-the-art YOLO for object detection and localization. Below, each approach is tackled thoroughly.

### I) Computer Vision Facial Feature Extraction and Machine Learning Classification:

The motivation behind this approach is a similar approach taken by MIT for gaze region estimation. Fridman, et al, used facial features to detect the gaze of the driver. We decided to follow a similar approach for feature extraction for our problem.

The pipeline for this method is: 1) face detection and alignment 2) feature extraction and selection 3) Classification and decision pruning. Below is a discussion for each step.

#### A) Face detection and alignment:

For face detection, we used the face detector implemented in DLib as it uses a Histogram of Oriented Gradients (HOG) combined with a linear Support Vector Machine (SVM) Classifier, and then an image pyramid, and sliding window detection scheme.

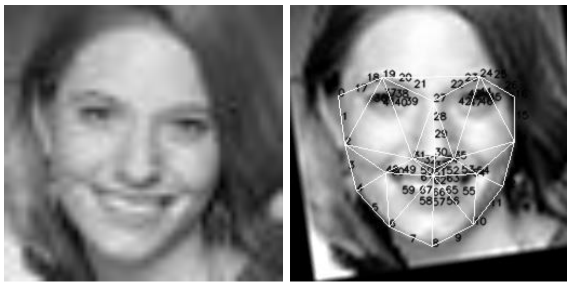
The reason behind our choice is that the performance of this detector has much lower false alarms rates than the widely-used default face detector available in OpenCV. In fact any false alarms will greatly affect our feature extraction, therefore, the DLib face detector fits to our purposes. After face detection, we used the face aligner from imutils so that all our input data is aligned to the same face orientation.

#### B) Feature Extraction and Selection:

We decided to use the 68-point Multi-PIE facial landmark mark-up scheme used in the iBUG 300-W dataset. There are 3 main reasons for our choice of this algorithm: (1) its robustness in getting correct facial features (2) it is robust to partial occlusion and self-occlusion and (3) its running-time is significantly faster than the 30 fps rate of incoming images. These landmarks include parts of the nose, upper edge of the eyebrows, outer and inner lips, jawline, and all parts in and around the eye. The selected landmarks are shown as numbered labeled in the fig below. Therefore, the first 136 elements in the feature vector are the (x, y) positions of the 68 points.

Moreover, 19 points out of the 68 are selected through recursive feature elimination. A Delaunay triangulation is computed over these 19 points and the three angles of each of the resulting triangles are added to the feature vector.

The result of this image processing and feature selection is shown in the fig below.



*a) Original image* *b) alignment & features*

C) Classification and decision pruning
We used a random forest classifier, with Sci-Kit-learn implementation for classification using the resulting feature vectors. The classifier has 550 trees. The classifier generates a set of probabilities for each class where probabilities are computed as the mean predicted class probabilities of the trees in the forest.

II) CNNs and MobileNetV3:

The second solution is using CNNs and MobileNetV3. In this solution we used Keras and CNNs to build a model that is trained on the same dataset explained above. Then, we used MobileNetV3 architecture to have a pretrained model and then we tried different combinations of this, by using different optimizers, adding or deleting Dense layers or adding and deleting CNN layers. In the experimentation part below, we explain the different models alongside their results.

In this solution there are three components, the model that we are training on the data set using CNNs (FacialExpressionDetector.ipnyb) which has the code of the model and importing the dataset. This file exports to us the weights corresponding to the best model (bestModel.h5). The third and last part of this solution is the (videoDetector.py) which is a python file that uses OpenCV to take a real time video stream of the user and draw a box around his/her face.

MobileNetV3 is a convolutional neural network that is tuned to mobile phone CPUs through a combination of hardware-aware network architecture search (NAS) complemented by the NetAdapt algorithm, and then subsequently improved through novel architecture advances. Advances include (1) complementary search techniques, (2) new efficient versions of nonlinearities practical for the mobile setting, (3) new efficient network design.

First, Explaining Facial Expression Detector :

We connect to Kaggle API to retrieve the dataset directly and quickly. The first step in doing so is installing Kaggle API. Then we enter the API key associated to the account on Kaggle that is saved in a kaggle.json file, after this authorization step we are connected directly to the platform and can get datasets directly from there. Then we get our desired dataset and unzip it. After that we import the needed libraries and build our model, which is a pretrained model importing MobileNetV3 model and adding a final 7-nodes output layer with activation softmax to detect the seven classes we are having. Afterwards, we start building our model, in the case of the fully CNN architecture we build our own architecture. Afterwards in the case of MobileNetV3 we call the pretrained model and then add layers to it as shown in the following code.

The input to our model is a 224x224x3 set of images and goes through a series of convolution (CNN) layers which are 13 convolution layers with different activation functions and at the end we have a seven-node output layer with activation softmax. The total number of trainable parameters is 351239 which is quite sufficient to the amount of data we are having.

However, in our fully connected CNN we changed the size of the pictures and made it 48x48x3 to not stuck in an overfitting hole. After building the model we compile it with "Adam" optimizer. We tried other optimizers as shown in the experimentation section; however, Adam optimizer showed the best results.

Second, Data Augmentation:

It was crucial to augment more images both to the training and the validation data. We used

ImageDataGenerator of Keras and we augmented the images to each of the seven classes.

Important Checkpointing and Early Stopping:

We then train our model for 30 epochs while enabling checkpointing and early stopping. Checkpointing allows us to save the state of the model for every specific number of epochs so that we can return to it if we want. Early stopping which is also available in Keras allows us to stop training if there is no change in our desired metric. We measure the change in the validation accuracy and if there is no increase in the validation accuracy with a minimum of 0.01 for 8 consecutive epochs then the model stops the training and at each checkpoint it saves its weights to bestModel.h5 if the model at this epoch had an increase in the validation accuracy.

At the beginning we tried to run the model for 100 epochs but we found that this is not needed at all, because the Early stopping used to stop after 15-20 epochs, so we used 30 epochs as the standard for all the models and experiments afterwards. However, this is the result of training the model for 100 epochs.

After the training we save the weights of the bestModel.h5 so that it can be used in the video detector to extract emotions from Realtime video of the user.

Afterwards, we plot the accuracy vs the validation accuracy and the loss vs the validation loss to see the performance of the model.

### III) YOLO:

YOLO is an algorithm that detects and recognizes various objects in the image. It is done as a regression problem and gives class probabilities for the detected image. YOLO uses bounding boxes to detect the width, height, center, and class of objects.

First, Data Labeling:

The dataset we used was mainly for classification. Yolo algorithm needs YOLO format data, which is basically ratios of the center of the bounding box, its width, and height. To tackle this problem, we implemented a python script to label the data. First it applied detection for the face in the image using DeeLib pretrained model. Then, it located the bounding box in the image and generate a text file with the class of the face and the position of the bounding boxThis generated the needed input data for YOLO.

Second, Model Training:

We used darknet version 4 package for implementing the algorithm. As advised in this framework, we downloaded yolov4-tiny-custom.cfg from darknet/cfg directory and applied some changes to it where we changed the batches to 64, subdivisions to 16, maximum batches to 30000 and adjusted the number of classes and filters that will be applied. The model was trained until the loss vanished constantly below 0.3.

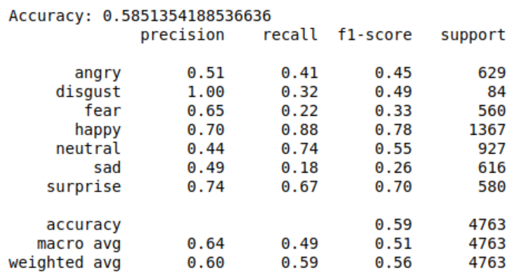
After that the model was trained and we tested the model on a live webcam.

## V. Results and analysis

We divided the dataset to 85 % for training and 15% (5380) images for test.

### I) Computer Vision Facial Feature Extraction and Machine Learning Classification:

After testing the detector, the results yielded an accuracy of around 60%. A complete report of each class accuracy is shown below:

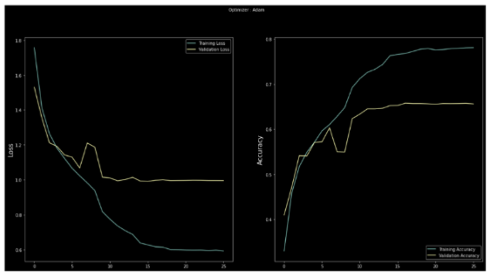


```
Accuracy: 0.5851354188536636
              precision    recall  f1-score   support

       angry       0.51      0.41      0.45       629
     disgust       1.00      0.32      0.49        84
        fear       0.65      0.22      0.33       560
       happy       0.70      0.88      0.78      1367
     neutral       0.44      0.74      0.55       927
         sad       0.49      0.18      0.26       616
    surprise       0.74      0.67      0.70       580

    accuracy                           0.59      4763
   macro avg       0.64      0.49      0.51      4763
weighted avg       0.60      0.59      0.56      4763
```

### II) CNNs and MobileNetV3

Model 1 (Fully CNN without MobileNet):

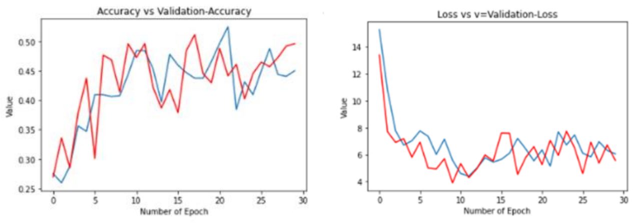
The first CNN architecture we implemented was based on extensive search in the literature for facial detection model architecture. After tuning the hyperparameters, we had results of this part was that we reached an average accuracy of around **78%**, which is the highest we got, as can be seen from the below figure and this was the best model out of the different CNN models followed below.



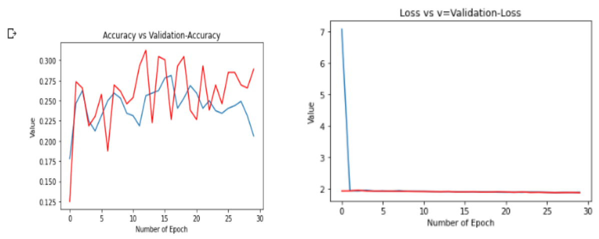
Results of the first CNN architecture

## Model 2 (MobileNet)

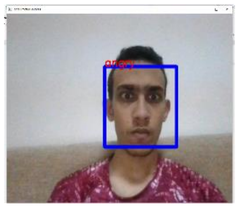
Original MobileNetV3 followed by a seven nodes output layer yielded accuracy around 45 – 47%, shown below:



When we added a dense layer with 64 nodes however, the number of trainable parameters increased tremendously and thus this led to overfitting due to the scarce of the data points we are having.

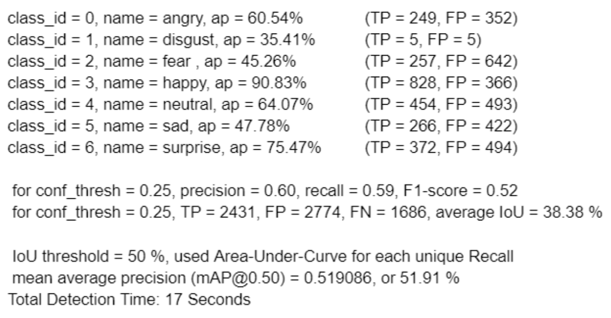


Video Detector Results:



## III) YOLO Results

After running the YOLO detector on the test set, it yielded the following results:



```
class_id = 0, name = angry, ap = 60.54%          (TP = 249, FP = 352)
class_id = 1, name = disgust, ap = 35.41%        (TP = 5, FP = 5)
class_id = 2, name = fear , ap = 45.26%          (TP = 257, FP = 642)
class_id = 3, name = happy, ap = 90.83%          (TP = 828, FP = 366)
class_id = 4, name = neutral, ap = 64.07%        (TP = 454, FP = 493)
class_id = 5, name = sad, ap = 47.78%            (TP = 266, FP = 422)
class_id = 6, name = surprise, ap = 75.47%       (TP = 372, FP = 494)

 for conf_thresh = 0.25, precision = 0.60, recall = 0.59, F1-score = 0.52
 for conf_thresh = 0.25, TP = 2431, FP = 2774, FN = 1686, average IoU = 38.38 %

 IoU threshold = 50 %, used Area-Under-Curve for each unique Recall
 mean average precision (mAP@0.50) = 0.519086, or 51.91 %
Total Detection Time: 17 Seconds
```

## VI. Development

Our recommendations are to use more clean datasets because clean datasets give better results. It is also better to hyper tune the parameters. Also, it is recommended to try different architectures for better results.

## VII. Conclusion

The results of the three solutions were very promising and they achieved very high accuracy.

## VIII. References

1. C. Darwin, The Expression of the Emotions in Man and Animals, John Murray, London, UK, 1872.
2. o1anuraganand. (2022, April 9). Emotion classification: CNN using Keras. Kaggle. Retrieved May 19, 2022, from https://www.kaggle.com/code/o1anuraganand/emotion-classification-cnn-using-keras
3. C. Sagonas, G. Tzimiropoulos, S. Zafeiriou, and M. Pantic, "300 faces in-the-wild challenge: The first facial landmark localization challenge," in Computer Vision Workshops (ICCVW), 2013 IEEE International Conference on. IEEE, 2013, pp. 397–403.
4. Face expression recognition dataset. (2022). Retrieved 19 May 2022, from https://www.kaggle.com/datasets/jonathanoheix/face-expression-recognition-dataset
5. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion,O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vander-plas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duch-esnay, "Scikit-learn: Machine learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011.