Facial Expression Detection **Ping Li, Dawson Geist, Lucy Lee\*, James Ramsay\***

Project is shared between ML and Data Mining class

\* Members of our data mining team

**Abstract**

In the last couple of decades interest in computer vision has drastically increased. As we can see in our everyday lives, computer vision is in our social media platforms, and even our cell phone cameras that allow us to tag those in our pictures using facial recognition. Automated facial imaging is now being used in research for automated self-driving vehicles, animation, surveillance and many other applications. Facial detection is just a stepping stone into the world of computer vision. This project will dive into computer vision to detect facial expressions using the Dlib and YOLO (You Only Look Once) algorithms. Dlib is an open source toolkit that contains algorithms that are widely used in Artificial Intelligence (AI), robotics, and even cell phones. YOLO is an open source real-time object detection algorithm capable of detecting objects in images, as well as videos, with high accuracy. We will compare these algorithms in detecting facial expressions in images.

**Introduction**

The goal of our project is to extrapolate emotion from facial expressions. With applications in multiple areas such as Law Enforcement, Business, and Developmental Psychology, effective and accurate emotion detection can play a major role in our society’s future. Some of the prospected applications of emotion detection include aiding in interrogations. By giving investigators the sentiment of a suspect’s response, they will have another metric that they can use to help them evaluate how truthful the suspect is being. Another application is in business, where companies can measure, in real time, the perceived response from an audience. This will provide actionable insights on the effectiveness of presentations, product launches, and business meetings.

The current solutions for emotion detection utilize Neural Networks to classify the emotion a person is feeling by interpreting the facial expression they are making. This is done by training a neural network with thousands of labeled images of facial expressions that correlate to different emotions.

This approach is limited by the fact that it is trained on static images of facial expressions. In many real-world scenarios, it is more accurate to measure a response to a stimulus by looking at the modular movements that piece together to form a facial expression. These are called action units, and looking at emotion as a composition of action units can provide a more robust, and expandable process for emotion classification.

Our project can be seen as two major components; analysis of facial expressions, and the neural network. We first extract the facial features (eyes, nose, eyebrows, lips, shape of head) from pictures of people with facial expressions related to six emotions (Happy, Sad, Surprise, Disgust, Fear, Anger, Neutral). These are then used to create feature vectors, which provide details like distance between the lips and distance between the eye and eyebrow. These feature vectors are then processed by our Neural Network which then provides the resulting emotion classification. By using facial features it will be easier to expand on our project and classify action units.

The goal for our project is to produce a neural network for emotion detection. With this model we hope to achieve similar or better results than open-source emotion detection python packages. Finally we hope to build a demo application that will classify emotion in real time using only the video from an integrated webcam.

**List of Tasks and Goals**

1. Apply Haar cascade classifiers on our data to detect the bounding boxes for the faces in the training images (\*Required for live demo)
2. Approximate the facial landmark coordinates for our test data set using Dlib
3. Construct our feature vectors based on the paper <Facial Expression Recognition using Facial Landmark Detection and Feature Extraction via Neural Networks>
4. Build the Neural Network as defined in the paper mentioned above
5. Prepare the input vectors for our training dataset
6. Train the dataset using the parameters mentioned in the paper as a baseline
7. Tune our model parameters if needed
8. Test our model on our test dataset
9. Evaluate our findings/ Visualize our data
10. Prepare live demo (if time permits)

**Data Set**

The data set is from kaggle.com and consists of 48x48 pixel grayscale images of faces. The faces have been automatically registered so that the face is more or less centered and occupies about the same amount of space in each image. The training set consists of 28,709 examples with the test set consisting of 3,589 examples. There are seven facial expression categories: (0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral). This dataset was prepared by Pierre-Luc Carrier and Aaron Courville, as part of an ongoing research project.

* <https://www.kaggle.com/competitions/challenges-in-representation-learning-facial-expression-recognition-challenge/data>

**Methods and Models**

Facial Detection: Since the data we are using is already limited to our Region of Interest(RoI), no future preprocessing will be needed. If we use a random image with noise, Haar cascade classifiers or YOLO facial detection can be used. Low Pass 3x3 Gaussian filter can also be considered to remove noise and sharpen the image. Regarding data transformation, we need to convert the csv file into matrix format using np.array.

Facial Landmark Detection: We will use two models for facial landmark detection; Dlib and YOLO. Dlib is a library that can detect 68 different facial landmarks. It uses the classic Histogram of Oriented Gradients(HOG) feature combined with a linear classifier, an image pyramid, and a slide window detection scheme. We can also get a pre-trained model file. YOLO was first introduced in 2015. It treats object detection as a regression problem rather than a classification problem. We plan to use both models and compare the final results. Before implementation, we will need to understand their structure and basic algorithms.

Feature Extraction: With the feature points (landmarks) obtained, we should decide the input feature vectors. We will use euclidean distance vectors for this. Also, we need distance vectors within each landmark, such as lip height and left/right eye height. So, there will be two sets of feature vectors.

Emotion Classification: Using the feature vectors, we will implement a Multi-Layer Perceptron(MLP) neural network, referring to the paper, <Facial Expression Recognition using Facial Landmark Detection and Feature Extraction via Neural Networks>. It consists of one input layer, two hidden layers, and one output layer.

**Assessment**

We will use a confusion matrix and assess the result based on the success rates. >98%: very well, >95%: relatively well, >90%:good, and >85%: moderate. We will also look into the false positive rate per emotion. Furthermore, we intend to compare the results from Dlib and YOLO. Emotions are subjective, so we expect some gray area between a couple of emotions. Nevertheless, we expect to detect noticeable differences in intra-landmark distances in each emotion, such as eye height with sadness and anger.

**Results and Visualization**

The primary visualization we will be implementing is a confusion matrix for the different emotion classifications. This is important because it shows the effectiveness of our classification algorithm by showing the prediction accuracy for a given class. In addition to the confusion matrix we also want to look at the relationships between the feature vectors and the different emotion classification. This would include traces of the average feature vector values for each respective emotion classification. To assess the performance of our model versus models that are publicly available, we will provide a table that shows the average accuracy for each emotion as well as the average time to compute a classification. Finally, we will prepare a live demonstration for our model which will showcase our models ability to provide real-time classification of a group member using our laptop’s integrated webcam.

**Roles**

Dawson:

My role is to be principal developer for the duration of this project. It will be my responsibility to learn the various packages this project needs to function as well as piecing together the modular components into a fully functional application. I will maintain constant communication with various team members as they will provide the clarification on the functionality that each module of the project requires so that the project functions as expected. I will be responsible for unit testing the modules I develop so that the output is correct and what my team members expect.

Ping:

My role is to understand how YOLO works and how it will contribute to our project. We will use YOLO to detect faces, operate the face landmarks and classify the expression output. After I understand YOLO’s principles, we will implement this library source into our project databases and we will compare it with Dlibs. My other potential role is to prepare our project to be expanded to use AU to detect movie or motion pictures expression.

Lucy:

My role is to understand the mechanism of Dlib. A complete understanding of the method we are using will give us a sturdy construction since Dlib will be one of our essential methods in this project. I plan to look into the codes and related papers and help other teammates understand its principle. Also, looking into different examples of Dlib's use will help us utilize the method more flexibly.

James:

My role will be to understand the YOLO algorithm and utilizing our data set on the algorithm. I will be understanding the code behind the algorithm and what the processes are to detect facial expressions. I will test its accuracy and fine tune the algorithm. We will then compare the results from the YOLO algorithm to the results of the Dlib algorithm.

**Schedule**

| **Date** |  | **Tasks to be Completed** |
| --- | --- | --- |
| 2/20/23 |  | Access/Inspect DataSet |
| 3/1/23 |  | Understanding Dlib |
| 3/9/23 |  | Mid Project Presentation Rehearsal |
| 3/14/23 |  | Implementing Dlib/YOLO on input data |
| 3/21/23 |  | Mid Project Presentation |
| 3/28/23 |  | Model Built |
| 4/27/23 |  | Final Project Presentation Rehearsal |
| 5/2/23 |  | Final Project Presentation |

**Bibliography**

* <https://arxiv.org/pdf/1812.04510.pdf>
* <https://arxiv.org/abs/1506.02640>
* [https://www.ri.cmu.edu/pub\_files/2015/5/intraface\_final.pdf](https://www.ri.cmu.edu/pub_files/2015/5/intraface_final.pdf%C2%A0)
* <https://datagen.tech/guides/face-recognition/facial-landmarks/>
* <https://pyimagesearch.com/2017/04/03/facial-landmarks-dlib-opencv-python/>
* <https://towardsdatascience.com/yolo-you-only-look-once-3dbdbb608ec4>
* <https://www.datacamp.com/blog/yolo-object-detection-explained>
* <https://keras.io/api/>
* <https://github.com/phamquiluan/ResidualMaskingNetwork>

**Final Report**

**Introduction**

The goal of our machine learning project was to extrapolate emotion from labeled images. By using state of the art machine learning methods, we were able to analyze images and use the information we extracted to build a model that is capable of predicting the emotion class for that image based on the facial expressions the person in the image was making. The motivation behind this project came from the real-world applications that can benefit from this technology. Areas of research currently applying this technology include: Social Media, Self-Driving Vehicles, Animation, and Surveillance. The goal of this project is to create an Artificial Neural Network that can perform on par with, or exceed, current open source facial detection / emotion detection models.

**Research Paper**

To achieve our goal we needed to assess the current area of research so that we could have an idea of where to start, and what to achieve. We used a research paper titled Facial Expression Recognition using Facial Landmark Detection and Feature Extraction via Neural Networks by Fuzail Khan. In this paper was a detailed procedure for achieving our goal of Emotion Classification. The process included Facial Detection, Landmark Detection, Extraction of Features, and Finally Model Construction, Training, and predicting using the Feature Vectors as input. The model that Khan created was a Multilayer Perceptron that consisted of an input layer with an unknown amount of nodes (determined by the number of features extracted), followed by two densely connected hidden layers of sizes 100, and 500 respectively. These layers both used ReLu activation functions. Finally there was a seven node output layer, one node for each emotion. This output layer used a soft sigmoid activation function. The hyper parameters for this model included a learning rate of 0.5% and a dropout rate of 30% at the final hidden layer. This model utilized the Adam Optimizer with its default python configuration.

**Data Set**

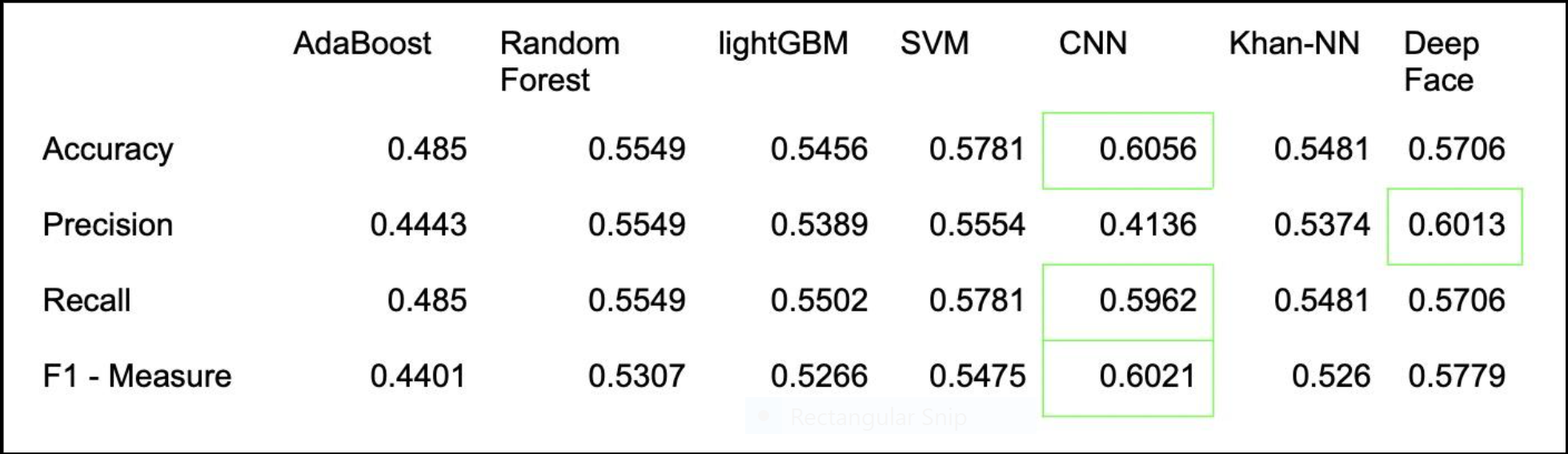
To train our model we used the Facial Emotion Detection (FER) 2013 dataset. This dataset contained roughly 36,000 labeled, black and white, 48x48 pixel images of people making facial expressions that correlated to the class label. This dataset had examples for the following 7 emotion classes; 0: Angry, 1: Disgust, 2: Fear, 3: Happy, 4: Sad, 5: Surprise, 6: Neutral. We split this dataset into a train set and a test set with a ratio of 70:30 train-test.

**Procedure**

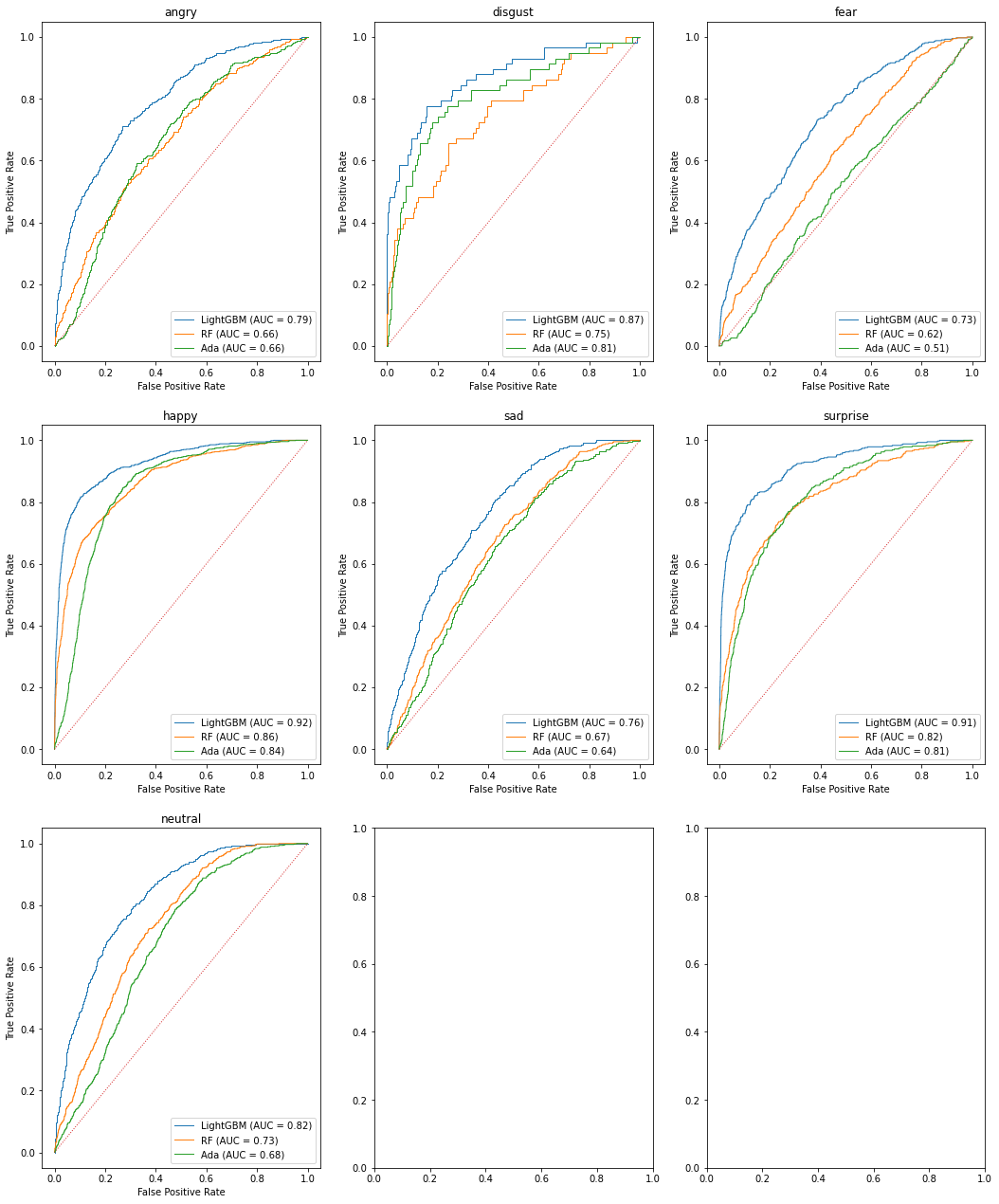
Using the Khan research paper as a guide we began working on each of the steps in the process.

1. Face Detection
   1. For face detection we tested multiple approaches to solving this, the main two being OpenCV which utilized Haar-Cascades and Dlib which used Histogram of Gradients (HoG). These two packages gave similar results and were used interchangeably to reduce noise by reducing our Area of Interest to only the pixels in the image that contained the subject’s face.
2. Landmark Extraction
   1. To do the landmark extraction we used a pretrained model that was part of the Dlib package. This model was based on a paper that used an ensemble of regression trees and gradient boosting to classify 68 points on the subject’s face that would detail the “shape” of the subject’s face. The landmarks provided coordinates for the outline subject’s eyes, nose, eyebrows, lips, and shape of the head
3. Feature Extraction
   1. Feature extraction was implemented by hand and required taking the locations of the 68 landmarks and finding the L2 distance between pairs of coordinates. These distances were strategically chosen to highlight the most important facial characteristics such as; The width and height of the mouth, the distance between the eyebrows and the eyes, the height of the eyes, the distance between the corner of the lips to the eyes, etc. These distances were to represent facial expressions such as Smiling vs. Frowning, Mouth Open vs. Closed, Eyebrows raised vs. lowered.
4. Model Selection, Training, and Prediction
   1. There were multiple models used to classify an emotion. The approaches we implemented included SVM, Random Forest, LightGBM, AdaBoosting, Khan-NN, and CNN, and DeepFace: VGG-Face (for evaluation)
      1. CNN Definition
         1. 1st double layer
            1. 64 filters each
            2. Batch normalization
            3. Max pooling (2,2)
         2. 2nd double layer
            1. 128 filters each
            2. Batch normalization
            3. Max pooling (2,2)
         3. 3rd double layer
            1. 256 filters each
            2. Batch normalization
            3. Max pooling (2,2)
         4. Fully connected layers
            1. Flatten
            2. Dense
            3. Batch normalization
            4. ReLu activation
            5. 20% dropout
            6. Dense
            7. Softmax activation
            8. Adam optimizer
5. Evaluation
   1. To assess our performance we used 5 metrics; Accuracy, Precision, Recall, F1-Score and ROC curve with AUC metric

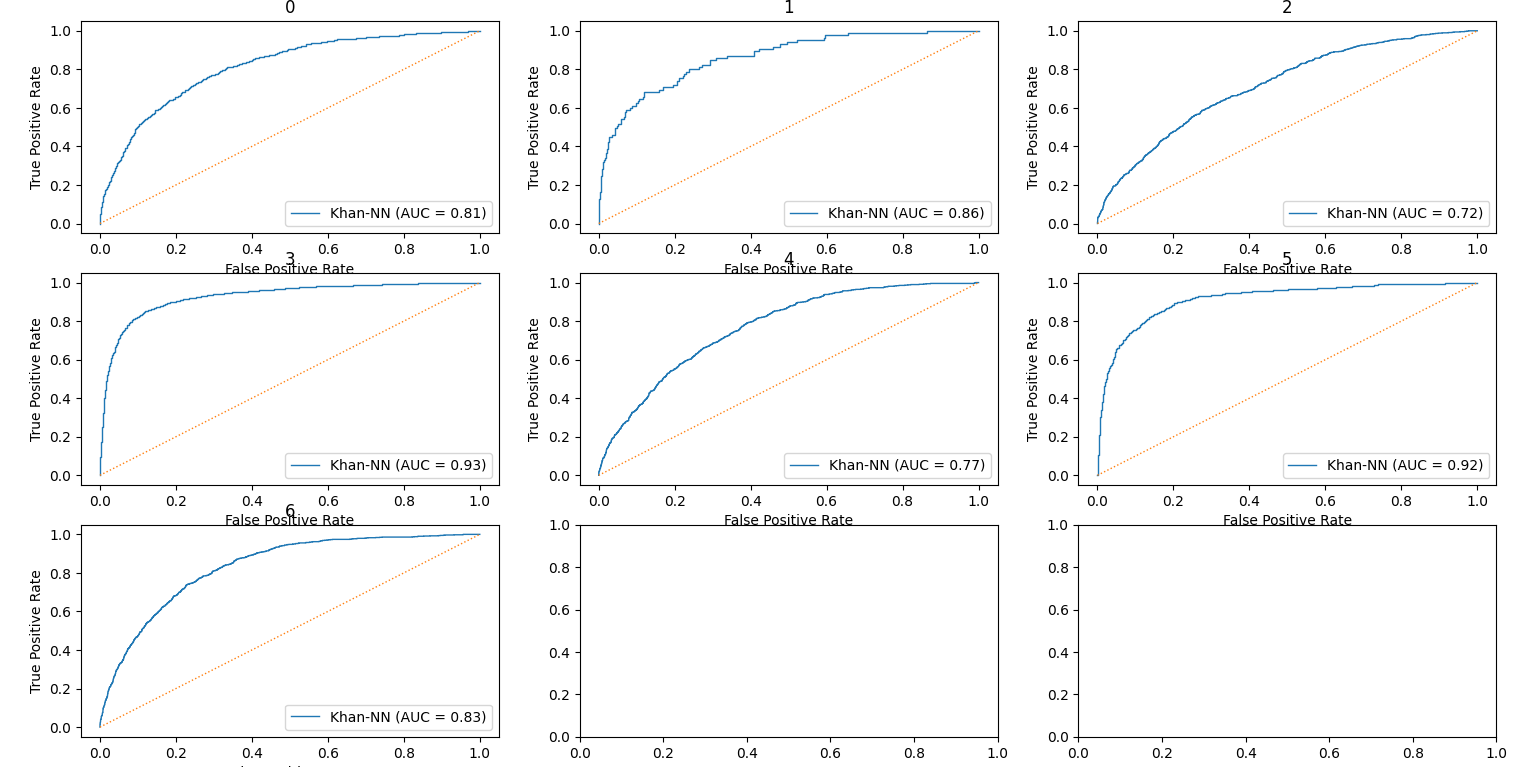
**Evaluation**



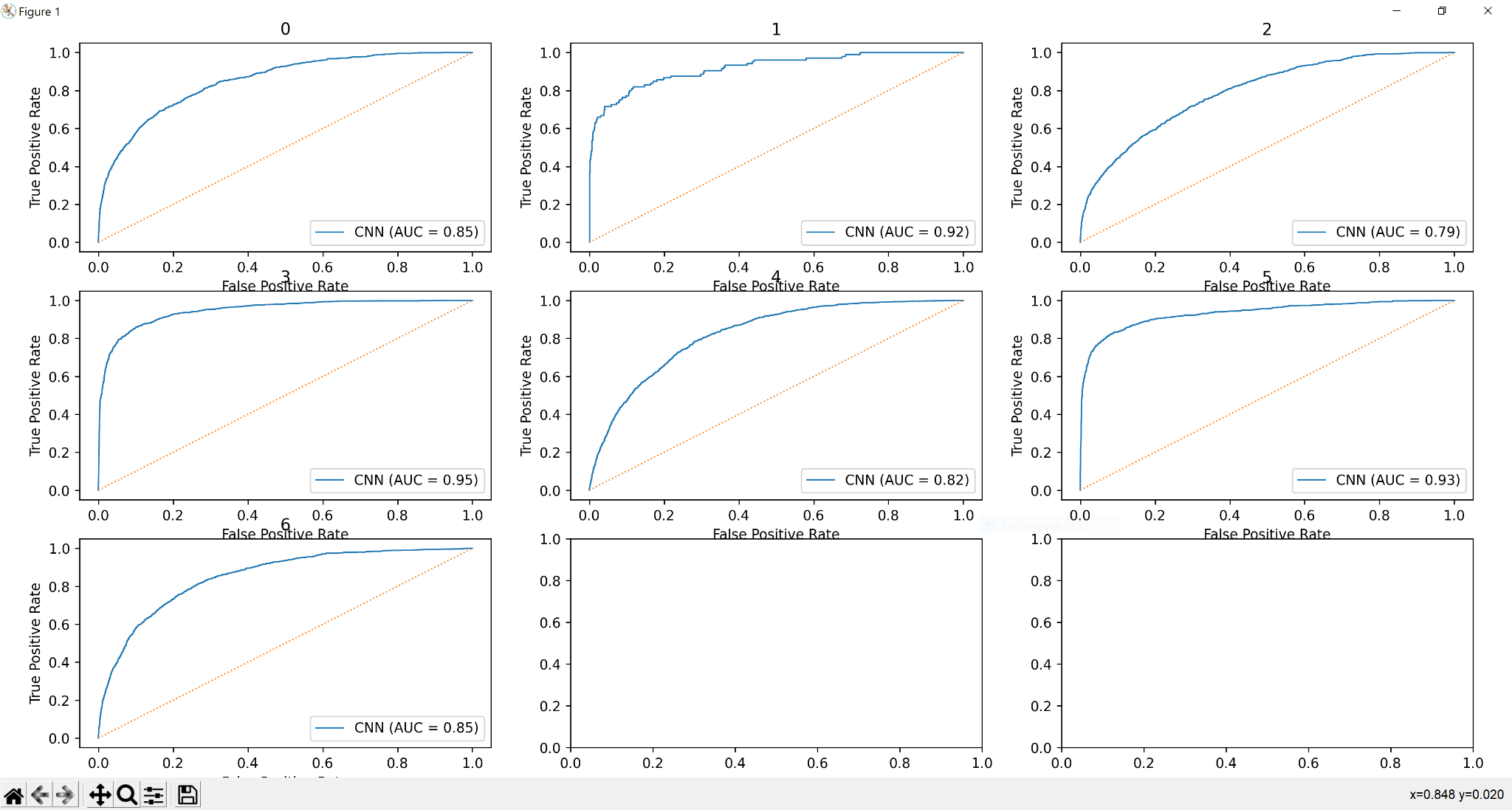
(ROC Below)

ROC-Curves

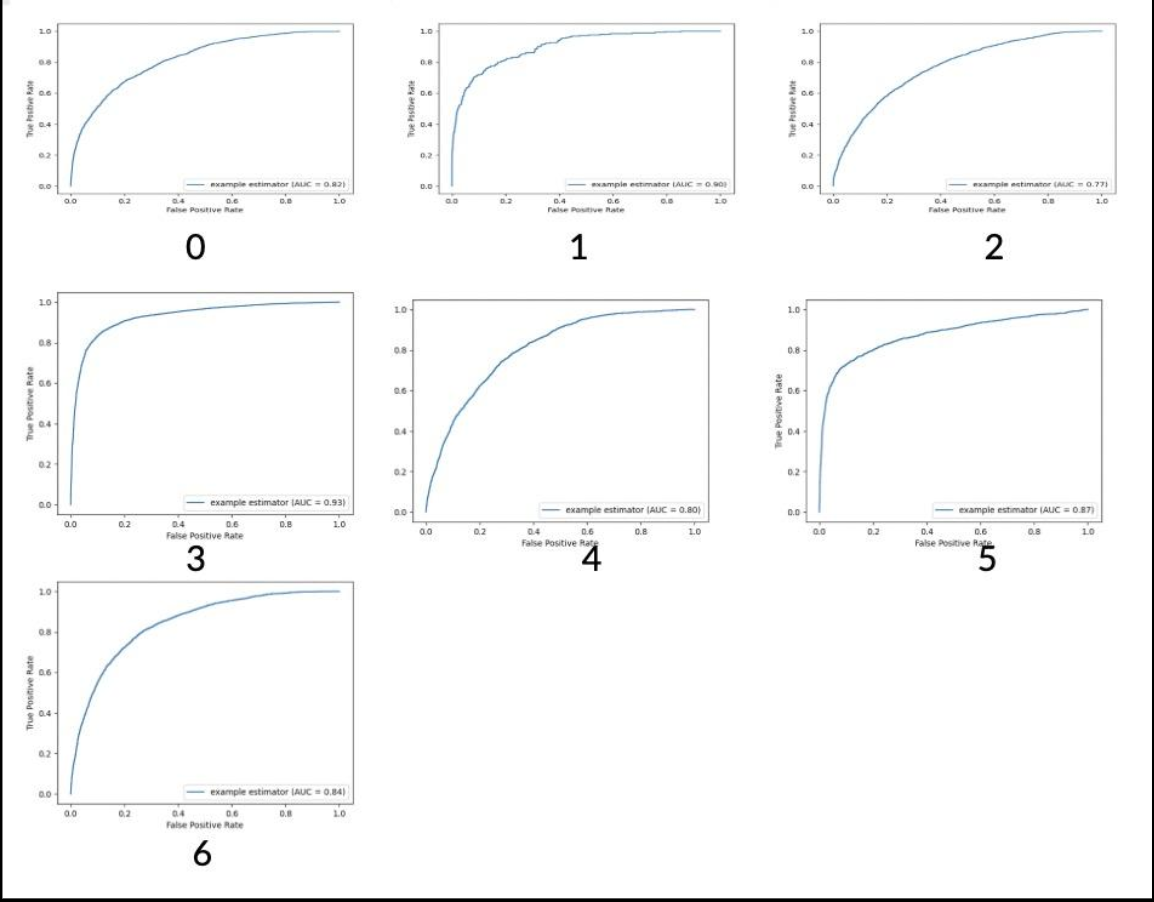
SkLearn Models



Khan-NN



CNN



VGG-Face

**Summary**

In conclusion we were able to achieve our goal of creating a comparable model that was able to outperform a state of the art model on the test set of our training data. Our results led to us creating a simple demonstration of real time emotion detection application that would analyze a video feed frame by frame and output the emotion classification along with its confidence score.

**Future Improvements**

Further improvements for this project could include migrating the code to C++ which would allow the computing of the feature vectors to occur faster and improve performance especially when it comes to frame rate of our demo application. In addition to migrating the code base, we could also try doing dimension reduction on our feature matrix so that only the most important features are available to our models to train on. Finally, to improve our results we can look towards creating a model that does the feature extraction and classification within the same model, much like how our CNN worked.

**References**

* https://arxiv.org/pdf/1812.04510.pdf
* <https://www.cv-foundation.org/openaccess/content_cvpr_2014/papers/Kazemi_One_Millisecond_Face_2014_CVPR_paper.pdf>
* <https://www.kaggle.com/datasets/msambare/fer2013>
* <http://dlib.net/>
* <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html>
* <https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html>
* <https://lightgbm.readthedocs.io/en/latest/pythonapi/lightgbm.LGBMClassifier.html>
* <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>