Facial Recognition Using YOLO

Jahnvi Arya (ja4158), Parijat Parimal (pp2206)

Introduction :

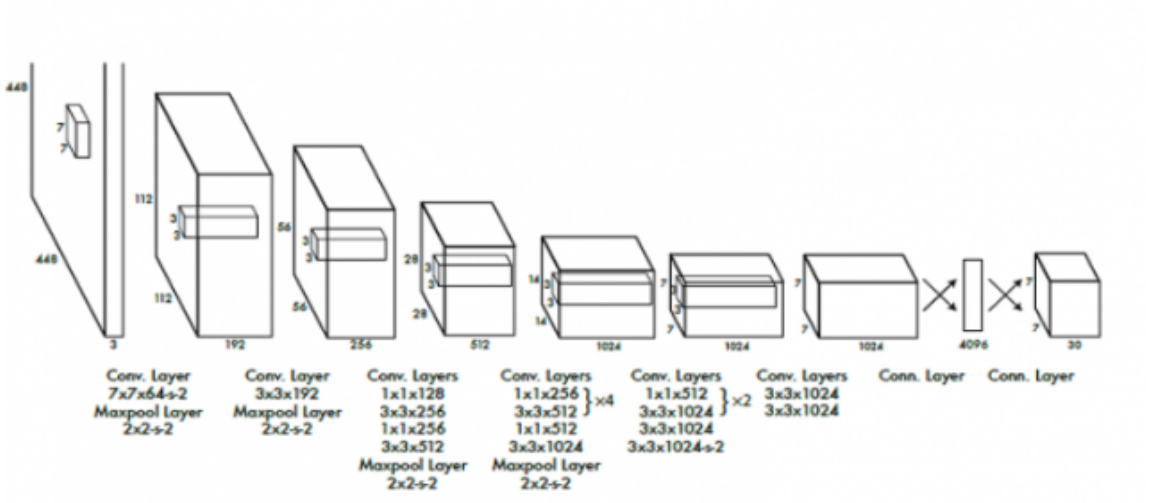
Face recognition is a method of identifying or verifying the identity of an individual using their face. There are two steps involved in face recognition. First is the detection and second is the recognition. It is important for us that the model can detect and recognize faces in minimum possible time from group photos as most of the use cases would require that. The most common use case that we think of is to use face recognition in cameras in public places to find missing people or wanted criminals. Another interesting application of face recognition is in video streaming platforms. For example, if you have used the x-ray feature in Amazon Prime videos, you could see every scene you can see the cast present in that particular scene. In all these scenarios, we need models that can handle multiple faces in single images and do that in a very short amount of time.

Objective :

In our experiment, we tried to explore the time as a metric since most of the algorithms give good performance on accuracy. We tried to improve time from detection to recognition of face without compromising the accuracy.

Background Work :

YOLO:



Object detection is one of the classical problems in computer vision where you work to recognize *what* and *where* – specifically what objects are inside a given image and also where they are in the image. The problem of object detection is more complex than classification, which also can recognize objects but doesn't indicate where the object is located in the image. In addition, classification doesn't work on images containing more than one object.

YOLO uses a totally different approach. YOLO is a clever convolutional neural network (CNN) for doing object detection in real-time. The algorithm applies a single neural network to the full image, and then divides the image into regions and predicts bounding boxes and probabilities for each region. These bounding boxes are weighted by the predicted probabilities.

YOLO is popular because it achieves high accuracy while also being able to run in real-time. The algorithm "only looks once" at the image in the sense that it requires only one forward propagation pass through the neural network to make predictions. After non-max suppression (which makes sure the object detection algorithm only detects each object once), it then outputs recognized objects together with the bounding boxes.

With YOLO, a single CNN simultaneously predicts multiple bounding boxes and class probabilities for those boxes. YOLO trains on full images and directly optimizes detection performance. This model has a number of benefits over other object detection methods:

- YOLO is extremely fast
- YOLO sees the entire image during training and test time so it implicitly encodes contextual information about classes as well as their appearance.
- YOLO learns generalizable representations of objects so that when trained on natural images and tested on artwork, the algorithm outperforms other top detection methods.

MTCNN :

MTCNN (Multi-task Cascaded Convolutional Neural Networks) is an algorithm consisting of 3 stages, which detects the bounding boxes of faces in an image along with their 5 Point Face Landmarks . Each stage gradually improves the detection results by passing it's inputs through a CNN, which returns candidate bounding boxes with their scores, followed by non max suppression.

In stage 1 the input image is scaled down multiple times to build an **image pyramid** and each scaled version of the image is passed through it's CNN. In stage 2 and 3 we extract image patches for each bounding box and resize them (**24x24** in stage 2 and **48x48** in stage 3) and forward them through the CNN of that stage. Besides bounding boxes and scores, stage 3 additionally computes **5 face landmarks points** for each bounding box.

VGG FACE: It is a very deep CNN architecture , which is learned on a large scale database, and is used as a feature extractor to extract the activation vector of the fully connected layer in the CNN architecture. Then, two types of subspace learning methods, namely, linear discriminant analysis (LDA) and whitening principal component analysis (WPCA), are respectively introduced to learn the subspace of the activation vectors for face recognition under multiple samples per subject and single sample per subject circumstances. The goals of applying subspace learning to the activation vectors are obtaining compact representation (dimensionality reduction) and performance improvement.

Approach:

Face recognition can be broadly divided into face detection and face recognition. Face detection is the part where a face is detected from an input image and the frame of the face is returned. Face recognition matches this face with the known faces to identify if the face belongs to a known person. In our approach (as shown in figure A1), we input an image to the object detection model, which if identifies a person, extracts the face of the person. This is done through MTCNN in model 1 and YOLO in model 2. Yolo is an object detection model, hence we are using a modified version of YOLO which detects only persons and extracts the faces of these people. We have used Yolo version 3, as we wanted to improve the inference time without compromising on the accuracy. MTCNN is a face detection algorithm and it directly gives us the faces from the images. These faces then go through a VGGface model, which returns the embeddings in these faces. We then use cosine similarities to match these embeddings with known embeddings. If the cosine similarity is greater than 0.5, we assume that the face matches that of the known person and hence we recognize the face. We then obtain the metrics to evaluate our models.

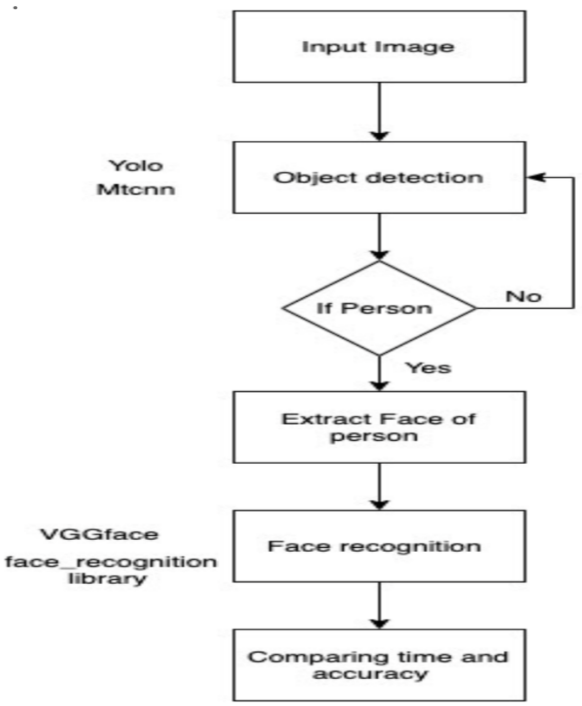


Figure A1 - Model flowchart

Experiment details:

- **Dataset**
  - Known data :- Einstein and Tesla
  - Unown Data :- Einstein, Tesla, image with multiple faces and some other unknown images.
- **Metrices**
  - Accuracy :- Correctly classifying faces
    - Precision
    - Recall
    - F1 score
  - Time :- Time to recognize faces
    - Cumulative time
    - Time per image
    - Time per face

Hardware specifications: 1 CPU with 8 GB RAM
Software specifications: The experiments were coded in python. Experiment demonstrations were performed on jupyter notebook. List of required frameworks and libraries has been mentioned in the readme file of the repository.

For the dataset, we have a set of known images which contains faces of people we know. And there is a set of images that may or may not contain faces of known people. For example, for known data, we have taken photos of Albert Einstein and Nikola Tesla. For unknown data, we have photos of Einstein, Tesla, Stephan Hawking and a group photo of many physicists.

We have the below metrics for evaluating our models. For the correctness of face recognition, we have used Precision, recall and F1 scores. Since most face recognition algorithms are very accurate these days, we were not aiming at getting better accuracy, rather our main focus was to get faster inference time, without much loss in accuracy.

For time, we have 3 metrics, one is the cumulative time required to recognize each face in the dataset. The other one is the average time to process an image and since we have images with multiple faces, the average time to process a face.

Observations And Performance evaluation:

These are the observations from our experiments. We can see that with the MTCNN model, we have two misclassified images, as when you notice the first two images, they belong to physicist Charles Wilson (shown in the figure A2), but they have been mislabeled as Tesla and Einstein. On the other hand, we can see that YOLO showed no misclassifications. Apart from these 2 models, we also tried a model based on python's built in library, called face_recognition. When we used the same images in this model, we could get faces only from a few of these image files. Notably, we got an output of 0 images with the group images, hence it would not make concrete comments on this model.

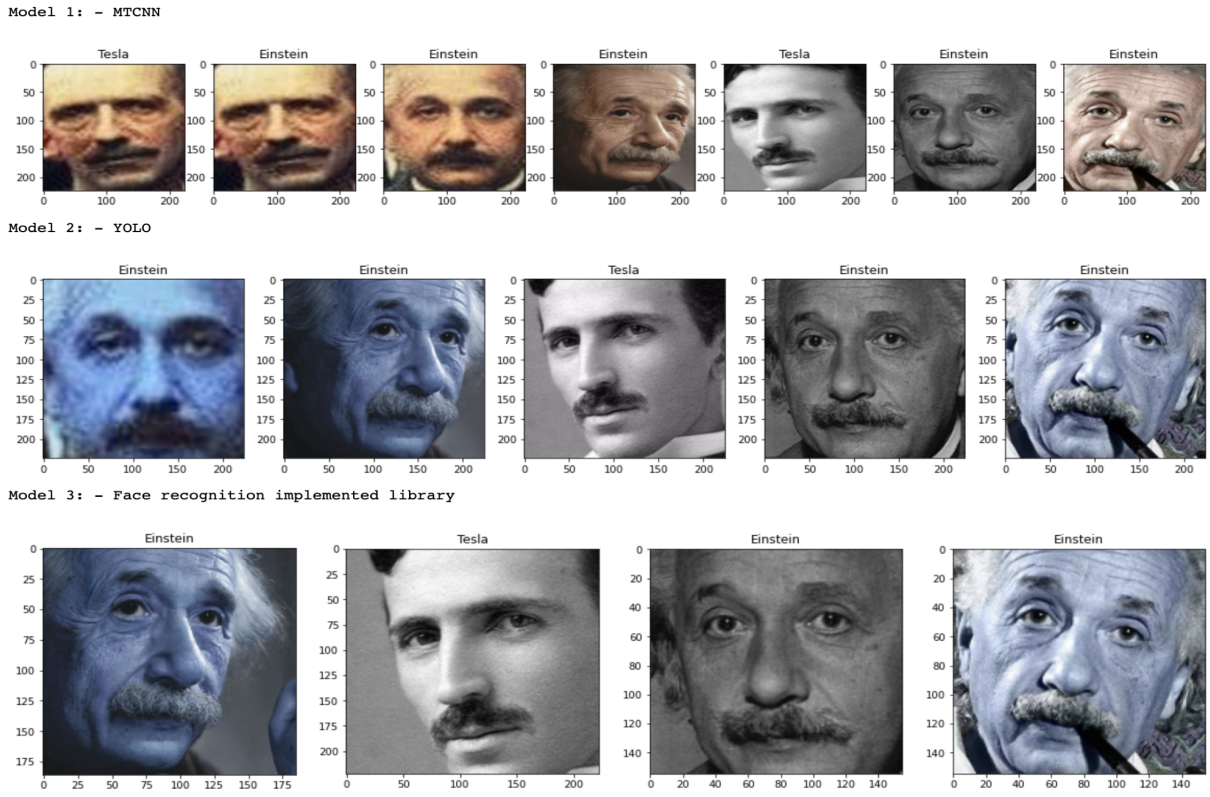


Figure A2 - Classifications with different models

Next we see the cumulative time of recognizing faces (shown in the figure A3). As we can see, model 2, which is the YOLO model, takes significantly less time to recognize faces. Another important observation is that the MTCNN model found one extra face from the group photo which was actually a photo of a tie. As stated earlier, model 3 did not work on the images with group photos, the graph for model 3 is very

small, but we can also see that it seems to be on the rise, compared to the YOLO model.

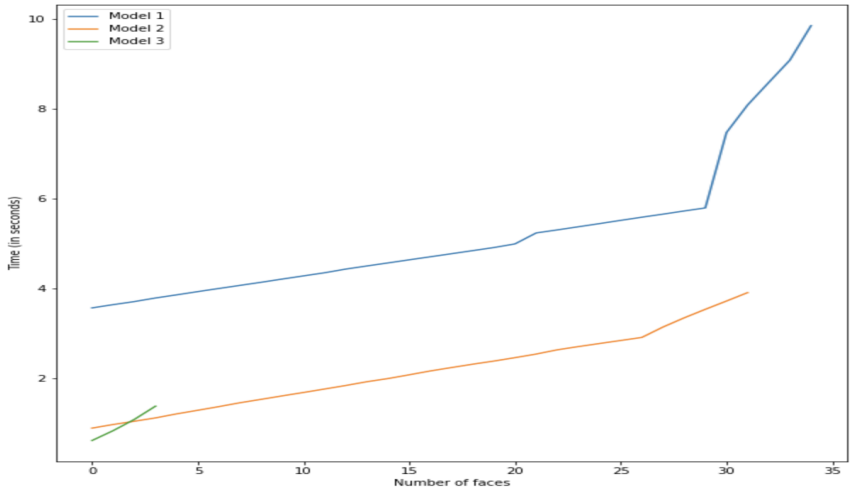


Figure A3 - Cumulative timeline of recognizing faces in different models. [Model 1 = MTCNN, Model 2 = YOLO, Model 3 = face_recognition library]

For accuracy related metrics (as shown in figure A4), we can see that the YOLO model actually outperforms MTCNN with a perfect F1 score of 1.0. And finally, we can see the contrast in time per file and time per face (as shown in figure A5). The MTCNN model took 1.23 seconds per file, whereas YOLO took 0.48 seconds. Also, MTCNN took 0.27 seconds per face, whereas YOLO took 0.10 seconds per face which is almost 1/3rd that of MTCNN. Also, we can see that model 3 outperforms YOLO in time per file but not not in time per face. But as stated earlier, we cannot make concrete claims on model 3.

```
Model 1: Precision = 0.7142857142857143, Recall = 1.0, F1 Score = 0.8333333333333333
Model 2: Precision = 1.0, Recall = 1.0, F1 Score = 1.0
Model 3: Precision = 1.0, Recall = 0.8, F1 Score = 0.888888888888889
```

Figure A4 - F1 scores for different models

```
Model 1: Time per file = 1.2316612303256989, Time per face = 0.2737024956279331
Model 2: Time per file = 0.4885392189025879, Time per face = 0.10856427086724176
Model 3: Time per file = 0.23007563749949136, Time per face = 0.23007563749949136
```

Figure A5 - Time metrics for different models

Conclusion:

Finally, to summarize, we can conclude that we can get approximately 3 times faster face recognition by using YOLO, compared to MTCNN which is a popular face detection algorithm. Also, we got better accuracy with YOLO, but we do not wish to make claims, as our tests were done on limited datasets.

For future work, we want to extend our experiments to darknet v4, which is a newer version of YOLO and other face recognition algorithms like cascade classifier and facenet. We also wanted to experiment with video inputs, to see if we get similar results with videos as we are getting with images.

References :

1. https://arxiv.org/pdf/1506.02640.pdf
2. https://arxiv.org/abs/1804.02767
3. https://www.youtube.com/watch?v=ArPaAX_PhIs&list=PLkDaE6sCZn6Gl29AoE31iwdVwSG-KnDzF
4. https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78

Github Repo :

https://github.com/parijatparimal29/faceRecognition