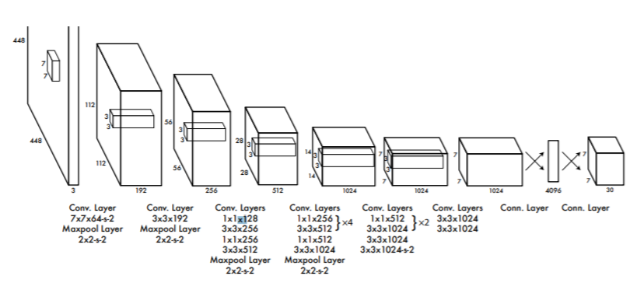
YOLOV3

Compared to the approach taken by object detection algorithms before YOLO, which repurpose classifiers to perform detection, YOLO proposes the use of an end-to-end [neural network](https://www.v7labs.com/blog/neural-network-architectures-guide) that makes predictions of bounding boxes and class probabilities all at once.

Following a fundamentally different approach to object detection, YOLO achieves state-of-the-art results beating other real-time object detection algorithms by a large margin.

Inspired by the GoogleNet architecture, YOLO’s architecture has a total of 24 convolutional layers with 2 fully connected layers at the end.



IoU

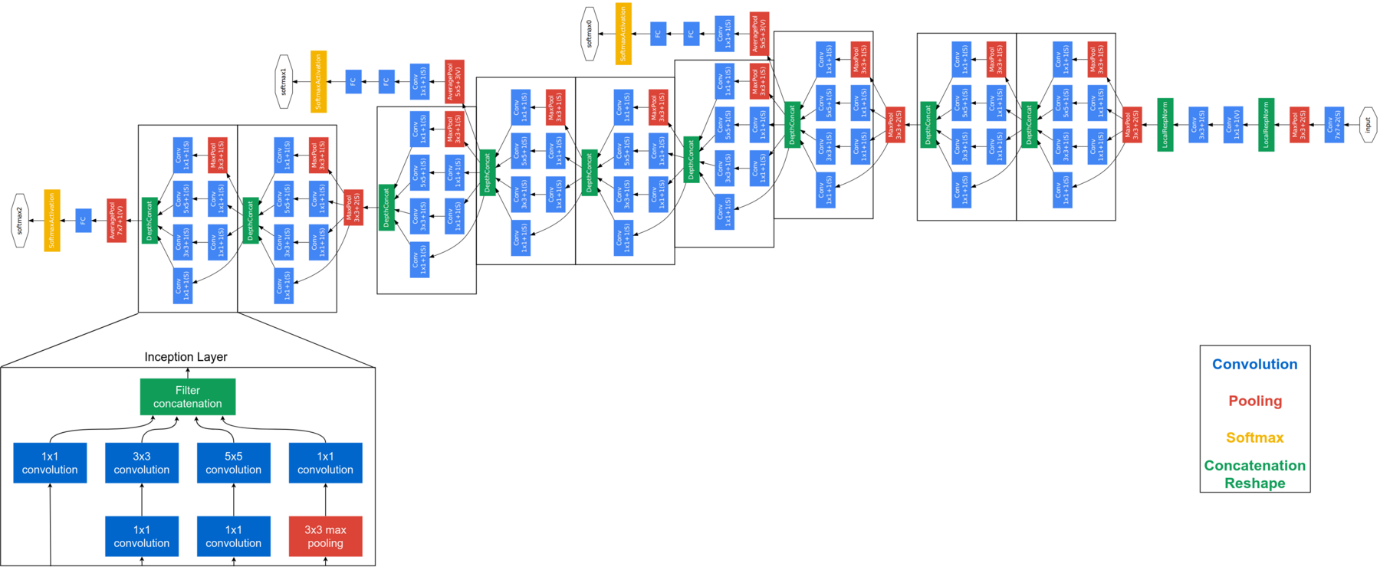
Intersection over Union is a popular metric to measure localization accuracy and calculate localization errors in object detection models.

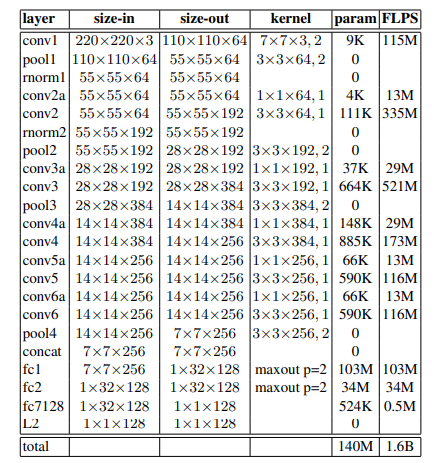
To calculate the IoU with the predictions and the ground truth, we first take the intersecting area between the bounding boxes for a particular prediction and the ground truth bounding boxes of the same area. Following this, we calculate the total area covered by the two bounding boxes—also known as the Union.

The intersection divided by the Union, gives us the ratio of the overlap to the total area, providing a good estimate of how close the bounding box is to the original prediction.

FACENET

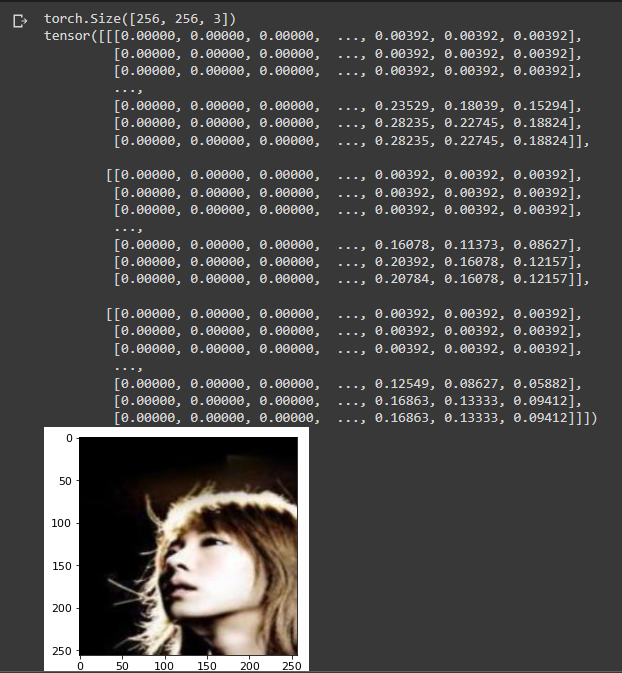
FaceNet is considered to be a state-of-art model developed by Google. It is based on the inception layer, explaining the complete architecture of FaceNet is beyond the scope of this blog. Given below is the architecture of FaceNet. FaceNet uses inception modules in blocks to reduce the number of trainable parameters. This model takes RGB images of 160×160 and generates an embedding of size 128 for an image. For this implementation, we will need a couple of extra functions. But before we feed the face image to FaceNet we need to extract the faces from the images.



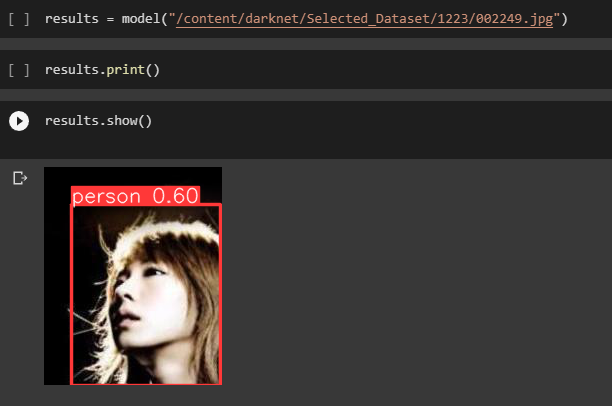


Experimental Results:

Image2vect

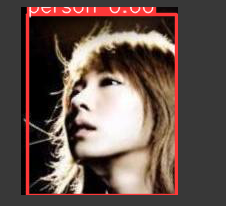


Here, this image is used as the test image to get the bounding box.



After initializing the model, using the image we get the bounding box on the face of the person in the image.

After getting the bounding box, we crop the image and then use it to get the vector of the cropped image.



imagefinder



YOLOV3 reference: <https://github.com/ultralytics/yolov3/releases>

https://www.v7labs.com/blog/yolo-object-detection#:~:text=The%20YOLO%20algorithm%20works%20by,of%20the%20object%20it%20contains.

FACENET reference: <https://github.com/timesler/facenet-pytorch>

https://www.geeksforgeeks.org/facenet-using-facial-recognition-system/