## MTCNN Model Architecture

MTCNN is a face detection model that uses a cascaded framework consisting of 3 models – P-Net, R-Net and O-Net. MTCNN will output the bounding boxes of the faces and the facial landmarks inside the bounding boxes.

### Models

The P-Net (Proposal Network) takes in the input image of 12x12 and obtains candidate windows and bounding box regression vectors. The bounding box regression vectors are used to optimize the candidate windows. To merge overlapped candidate windows, non-maximum suppression (NMS) is then used.

The R-Net (Refine Network) takes in the input image of 24x24 and rejects false candidate windows from the large number of candidate windows generated by P-Net. It is further optimized with bounding box regression and NMS.

The O-Net (Output Network) takes in the input image of 48x48 and further rejects false candidate windows similar to R-Net. It also outputs the positions of the 5 facial landmarks inside the bounding boxes.

### Parameters

There are 4 parameters to set in MTCNN class:

- weights_file: Model weights of P-Net, R-Net and O-Net

- min_face_size: minimum size of the face to detect, default is 20

- steps_threshold: Threshold for each model. Default is [0.6, 0.7, 0.7]

- scale_factor: Number of image pyramids used, default is 0.709

## MTCNN Source Code

https://github.com/ipazc/mtcnn/tree/master The source code by ipazc is an implementation of MTCNN as a python package. The source code allows the model weights of the model to be used for prediction.

However, there is no training code provided to change the model weights. I have done transfer learning using single-stage architecture like Resnet or Inception as it is relatively easy to call the model from TensorFlow library, add or edit new layers and retrain the model. However, with the multi-stage architecture of MTCNN, it is harder to add or edit new layers. I can only retrain 1 model at a time and not all 3 models together.

## MTCNN Training Code

I decided to find training code to modify the model weights and use it for prediction in ipazc's source code. I found an article that contains the training code to train P-Net. There is also a GitHub repository of training code by baomingwang linked in the article. I decided to try baomingwang's training code to see if I can replicate the result and use the model weights generated by the code for prediction in ipazc's source code.

### Problems Encountered

I tried to train the P-Net model only. I had to download the WIDERFace dataset to generate training data and train the model. However, the code is outdated as it uses TensorFlow version 1.0. I modified the code to allow version 1.0 compatibility. I also run the code without modifying anything to understand the code.

When I run the code to train the model, I encountered an error that prevented me from continuing with the repository as it was an error that I could not troubleshoot further. I decided to use the training code by the author of the article.

I generated training data and train the model without modifying anything. However, the code to update the model weights had errors and I decided to write alternative code to save the model weights separately as P-Net model. I retrieved the R-Net and O-Net model weights from FaceNet repository as ipazc mentioned the model weights the source code uses are from FaceNet but merged together as a single numpy file. I modified ipazc's source code to take in 3 separate model weights instead of 1. However, I got an error relating to the model weights.

```
ValueError: You called `set_weights(weights)` on layer "model" with a weight list of length 1, but the layer was expecting 13 weights. Provide
d weights: {'pnet': {'pnet': {'conv1': array([-0.11865927,  0...
```

I decided to write another code to update the model weights in mtcnn_weights.npy. It was successful as I managed to get a prediction on a sample image.

## Modifications

I decided to modify the parameters of the layers as I encountered errors when adding new layers. I changed the padding of all the layers from 'VALID' to 'SAME'. I also set 'relu' parameter of conv2 and conv3 to True.

For the parameters, if I increase min_face_size, the confidence rate increases. If R-Net and O-Net thresholds in steps_threshold increases, confidence rate increases. If scale factor increases, confidence rate increases.

Result right after training P-Net:

```
Running Pnet!
[array([[[[0.09913689, 0.19981617, 0.7198129 , 0.6677117 ]]]],
      dtype=float32), array([[[[0.37227347, 0.6277265 ]]]], dtype=float32)]
```

Result of model prediction:

```
[{'box': [276, 88, 51, 68], 'confidence': 0.9999830722808838, 'keypoints': {'left_eye': (291, 117), 'right_eye': (314, 114), 'nose': (303, 131), 'mouth_left':
(296, 142), 'mouth_right': (313, 141)}}]
```