# Follow Me

## Network Architecture:

| Layer (type) | Output Shape | Param # | Connected to |
|---|---|---|---|
| input_1 (InputLayer) | (None, 160, 160, 3) | 0 | |
| separable_conv2d_keras_1 (Separa | (None, 160, 160, 32) | 155 | input_1[0][0] |
| batch_normalization_1 (BatchNorm | (None, 160, 160, 32) | 128 | separable_conv2d_keras_1[0][0] |
| separable_conv2d_keras_2 (Separa | (None, 80, 80, 64) | 2400 | batch_normalization_1[0][0] |
| batch_normalization_2 (BatchNorm | (None, 80, 80, 64) | 256 | separable_conv2d_keras_2[0][0] |
| separable_conv2d_keras_3 (Separa | (None, 40, 40, 128) | 8896 | batch_normalization_2[0][0] |
| batch_normalization_3 (BatchNorm | (None, 40, 40, 128) | 512 | separable_conv2d_keras_3[0][0] |
| separable_conv2d_keras_4 (Separa | (None, 20, 20, 256) | 34176 | batch_normalization_3[0][0] |
| batch_normalization_4 (BatchNorm | (None, 20, 20, 256) | 1024 | separable_conv2d_keras_4[0][0] |
| conv2d_1 (Conv2D) | (None, 20, 20, 512) | 131584 | batch_normalization_4[0][0] |
| batch_normalization_5 (BatchNorm | (None, 20, 20, 512) | 2048 | conv2d_1[0][0] |
| bilinear_up_sampling2d_1 (Biline | (None, 40, 40, 512) | 0 | batch_normalization_5[0][0] |
| concatenate_1 (Concatenate) | (None, 40, 40, 640) | 0 | bilinear_up_sampling2d_1[0][0]<br>batch_normalization_3[0][0] |
| separable_conv2d_keras_5 (Separa | (None, 40, 40, 256) | 169856 | concatenate_1[0][0] |
| batch_normalization_6 (BatchNorm | (None, 40, 40, 256) | 1024 | separable_conv2d_keras_5[0][0] |
| bilinear_up_sampling2d_2 (Biline | (None, 80, 80, 256) | 0 | batch_normalization_6[0][0] |
| concatenate_2 (Concatenate) | (None, 80, 80, 320) | 0 | bilinear_up_sampling2d_2[0][0]<br>batch_normalization_2[0][0] |
| separable_conv2d_keras_6 (Separa | (None, 80, 80, 128) | 43968 | concatenate_2[0][0] |
| batch_normalization_7 (BatchNorm | (None, 80, 80, 128) | 512 | separable_conv2d_keras_6[0][0] |
| bilinear_up_sampling2d_3 (Biline | (None, 160, 160, 128) | 0 | batch_normalization_7[0][0] |
| concatenate_3 (Concatenate) | (None, 160, 160, 160) | 0 | bilinear_up_sampling2d_3[0][0]<br>batch_normalization_1[0][0] |
| separable_conv2d_keras_7 (Separa | (None, 160, 160, 64) | 11744 | concatenate_3[0][0] |
| batch_normalization_8 (BatchNorm | (None, 160, 160, 64) | 256 | separable_conv2d_keras_7[0][0] |
| conv2d_2 (Conv2D) | (None, 160, 160, 3) | 195 | batch_normalization_8[0][0] |

- Layer for the input with a filter of 32 then 3 layers of encoder with filters (64,128,256) then a 1x1 conv layer with filter (512) then 3 decoder layers with filters (256,128,64) with each decoder layer connected to an encoder layer that has half the number of filters ex : decoder layer with filter 256 is connected to encoder layer with filter 128.
- Encoder uses separable convolution which has 2 steps , 1 to convolute Over each of the input channels and produce an output channel with same width and height of the input channel using 1x1 convolution, this helps improve performance during runtime as it reduces number of parameters.
- Decoder uses Bilinear upsampling which can be used to increase the size of the image this is done to help us identify the location of the person as during the encoding layers spatial information is lost After that concatenation is done to integrate features together

- I have tried adding more layers to encoder and decoder but it didn't increase accuracy, probably due to limited data set that I have

- I tried stride of 1, 2, 3 for the encoder block, both strides of 1 and 3 gave bad accuracy and low performance respectively , so a stride of 2 seemed to be the most promising.

- The 1x1 conv layer retains the spatial information, and it connects encoders and decoders, I tried setting the filter same as the previous encoder layer but it didn't improve accuracy so I increased the filters and got better result.

Here are the parameters for the FCN:

learning_rate = 0.005

batch_size = 32

num_epochs = 20

steps_per_epoch = 140

validation_steps = 50

workers = 8

Increasing the number of epochs won't increase the accuracy since the val_loss and train_loss tends to flatten, also to avoid over-fitting for the data , I also used a small learning rate for that reason aswell.

Batch size was the default set and I didn't change.

This network can be used to identify other objects if we change the training and validation set. if we want to detect more than 3 objects the we need to change this variable:

```
num_classes = 3
```

to the number we desire, we will also need to add more layers and get a larger data set with the objects we want to recognize.

I started with an accuracy of 35% then 39% and finally got 40.008% then after applying final tweeks I managed to get 44.87% accuracy , I think I can get a better accuracy if I got more dataset to train my network on and more for validation