## Project: Follow Me

## Rubric Points

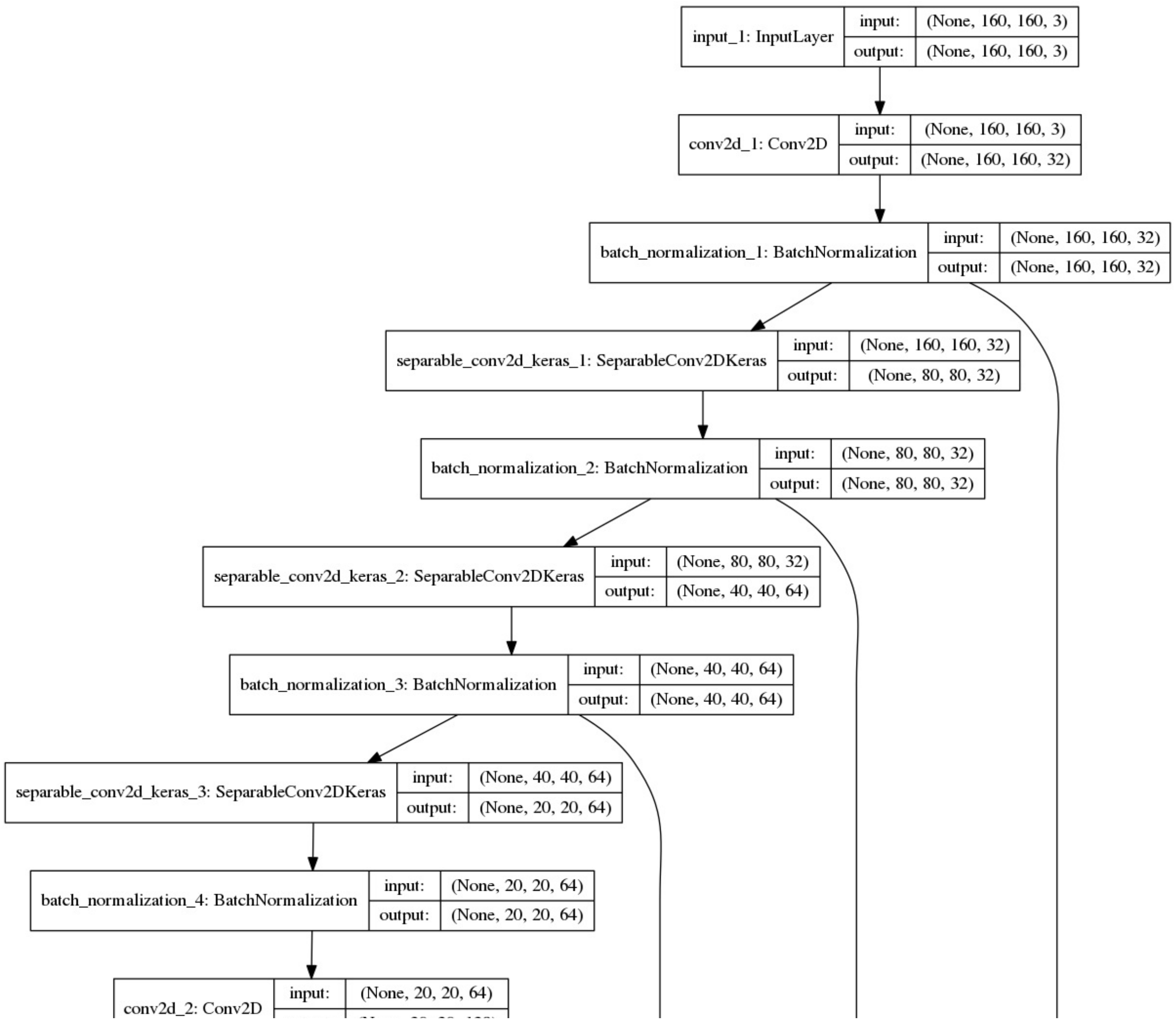
### Writeup / README

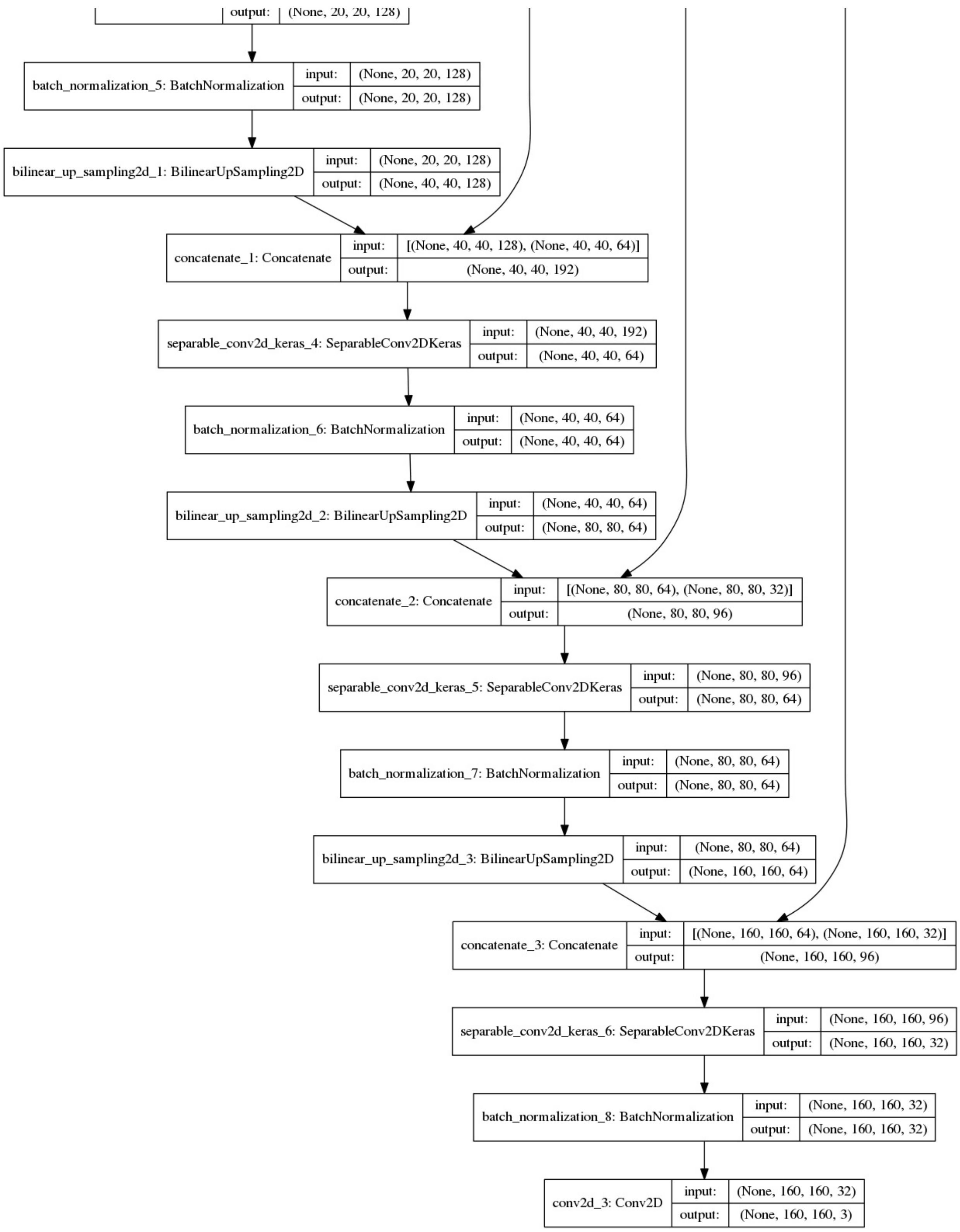
#### 1. Provide a write-up / README document including all rubric items addressed in a clear and concise manner. The document can be submitted either in either Markdown or a PDF format.

You're reading it!

#### 2. The write-up conveys the an understanding of the network architecture.

Here is the structure of my model.


input_1: InputLayer
input: (None, 160, 160, 3)
output: (None, 160, 160, 3)
conv2d_1: Conv2D
input: (None, 160, 160, 3)
output: (None, 160, 160, 32)
batch_normalization_1: BatchNormalization
input: (None, 160, 160, 32)
output: (None, 160, 160, 32)
separable_conv2d_keras_1: SeparableConv2DKeras
input: (None, 160, 160, 32)
output: (None, 80, 80, 32)
batch_normalization_2: BatchNormalization
input: (None, 80, 80, 32)
output: (None, 80, 80, 32)
separable_conv2d_keras_2: SeparableConv2DKeras
input: (None, 80, 80, 32)
output: (None, 40, 40, 64)
batch_normalization_3: BatchNormalization
input: (None, 40, 40, 64)
output: (None, 40, 40, 64)
separable_conv2d_keras_3: SeparableConv2DKeras
input: (None, 40, 40, 64)
output: (None, 20, 20, 64)
batch_normalization_4: BatchNormalization
input: (None, 20, 20, 64)
output: (None, 20, 20, 64)
conv2d_2: Conv2D
input: (None, 20, 20, 64)


output: (None, 20, 20, 128)
batch_normalization_5: BatchNormalization
input: (None, 20, 20, 128)
output: (None, 20, 20, 128)
bilinear_up_sampling2d_1: BilinearUpSampling2D
input: (None, 20, 20, 128)
output: (None, 40, 40, 128)
concatenate_1: Concatenate
input: [(None, 40, 40, 128), (None, 40, 40, 64)]
output: (None, 40, 40, 192)
separable_conv2d_keras_4: SeparableConv2DKeras
input: (None, 40, 40, 192)
output: (None, 40, 40, 64)
batch_normalization_6: BatchNormalization
input: (None, 40, 40, 64)
output: (None, 40, 40, 64)
bilinear_up_sampling2d_2: BilinearUpSampling2D
input: (None, 40, 40, 64)
output: (None, 80, 80, 64)
concatenate_2: Concatenate
input: [(None, 80, 80, 64), (None, 80, 80, 32)]
output: (None, 80, 80, 96)
separable_conv2d_keras_5: SeparableConv2DKeras
input: (None, 80, 80, 96)
output: (None, 80, 80, 64)
batch_normalization_7: BatchNormalization
input: (None, 80, 80, 64)
output: (None, 80, 80, 64)
bilinear_up_sampling2d_3: BilinearUpSampling2D
input: (None, 80, 80, 64)
output: (None, 160, 160, 64)
concatenate_3: Concatenate
input: [(None, 160, 160, 64), (None, 160, 160, 32)]
output: (None, 160, 160, 96)
separable_conv2d_keras_6: SeparableConv2DKeras
input: (None, 160, 160, 96)
output: (None, 160, 160, 32)
batch_normalization_8: BatchNormalization
input: (None, 160, 160, 32)
output: (None, 160, 160, 32)
conv2d_3: Conv2D
input: (None, 160, 160, 32)
output: (None, 160, 160, 3)

3 encoders and 3 decoders are used, with [32, 64, 64] filters of each encoder and [64, 64, 32] in each decoder and the 1x1 conv layer has 128 filters.

The encoders extract the useful messages from the image, and decoders upsample the input feature map and concatenates with encoder with the same size.

## 3. The write-up conveys the student's understanding of the parameters chosen for the the neural network.

```
def fcn_model(inputs, num_classes):
    inputs = conv2d_batchnorm(inputs, filters=32, kernel_size=1, strides=1)
    encoder1 = encoder_block(inputs,filters = 32,strides =2)
    encoder2 = encoder_block(encoder1,filters = 64,strides =2)
    encoder3 = encoder_block(encoder2,filters = 64,strides =2)
    onebyone_conv = conv2d_batchnorm(encoder3, filters = 128, kernel_size=1,strides=
    decoder1 = decoder_block(onebyone_conv,encoder2,filters = 64,up_factor=2)
    decoder2 = decoder_block(decoder1,encoder1,filters = 64,up_factor=2)
    decoder3 = decoder_block(decoder2,inputs,filters = 32,up_factor=2)
    x = decoder3
    return layers.Conv2D(num_classes, 1, activation='softmax', padding='same')(x)
```

The encoder is mush like a normal convolution network, but it uses separable conv nets. The layer scales down in width and height but becomes deeper . So I choose [32, 64, 64]. The kernel_size of convolution networks is 3 by default and I tried 2 but got a bad performance. I thought a stride of 2 and a kernel of 3 would cause some overlap and reutilize of the input. But strde=2, kernel_size=2 didn't show a more pleasant result.

The 1x1 conv layer retains the spatial information, and it connects encoders and decoders. I decide to set 128 filters for it.

The decoder is symmetric to encoder, the params like filters, strides, kernel size are exactly the same as their counterparts.

I tried many many different params and found 3 or 4 encoders and decoders gives the best result.

```
learning_rate = 0.008
batch_size = 32
num_epochs = 15
steps_per_epoch = 200
validation_steps = 50
workers = 4
```

As to the hyper-parameters, I tried to increase the batch_size and epochs, but it often goes to overfitting, especially when val_loss and train_loss are low. The val_loss would fluctuate. To avoid overfitting, I picked relatively small learning_rate=0.008 with decay=0.0003. The batch_size=32 and epochs=15, I think that's an ordinary value.

## 4. The student has a clear understanding and is able to identify the

## use of various techniques and concepts in network layers indicated by the write-up.

Here is my understanding of concepts and techniques used in my network.

- sperable convolution: The separable convolution is to convolute depthwise. The in_channel is equal to out_channel, with in_channel numbers of filters which convolute on each feature map.

- batch normalization: Batch normalization is to get the mean and stv of the minibatch images, then every image is subtracted by the mean and divided by stv. Batch normalization is used on input of every layer, making the input data normalized and more easy to train.

- encoder: The encoder uses the sperable convolution and batch normalization to extract infomation from the image.

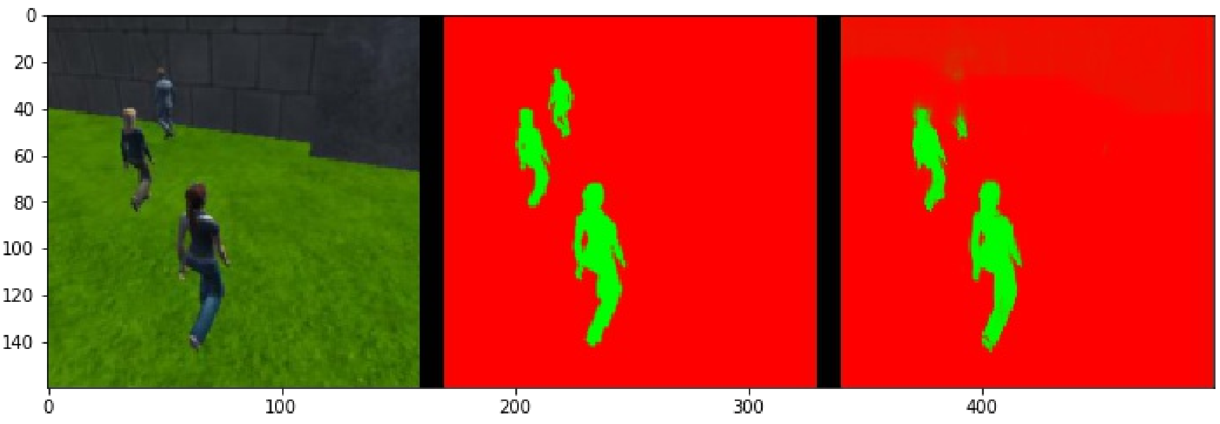
- 1x1 convolution: 1x1 conv is a convolution with kernel size of 1. It helps to reduce dimensionality(the number of parameters).

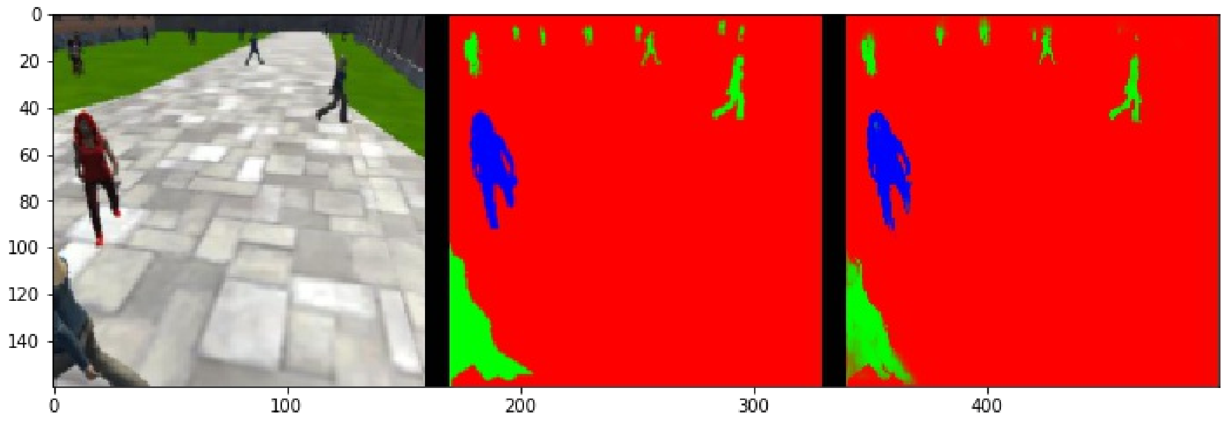
- upsampling: Upsampling is the opposite way to downsampling or pooling. It double or triple the size of input image.

- concatenation: Concatenation is used to put feature maps together. The feature maps must have the same height and width, so it can be concatenated.

- decoder: The decoder upsamples the smaller image, and concatenates with the larger image. The smaller image is from the previous layer and the larger one is from encoder. By combining thoese feature maps together, the result would retain much original pixel infomation.

## 5. The student has a clear understanding of image manipulation in the context of the project indicated by the write-up.





The original rgb image(left) is the input for traing, validation and testing. The label(middle) is processed by the DroneSim Simulator which gives a semantic segmentation result. Our semantic segmentation model is a kind of FCN(Fully Convolutional Network), by classifying every pixel into different classes. In our case, there are 3 classes: hero(blue), people(green) and the background(red). A softmax activation is used to determine which class the pixel belongs to.

### 6. The student displays a solid understanding of the limitations to the neural network with the given data chosen for various follow-me scenarios which are conveyed in the write-up.

I got a final_score = 0.444490934687, final_Iou = 0.581808046281, which passes the rubric score. The drawback of the neural network is when detecting and classifying other people. It's not so accurate as hero and background. When I try follow-me using this model, the quadroter can detect the hero accurately, but occasionally it recognizes objects of dark color as the hero then immediately loses it. When it finds the hero, it keeps track of the hero and never lose.

I think the pixel-wise classification is not so accurate, because the model is somewhat simple, it has only 43107 trainable params, which would be more generous than precise. No pooling layer used and the number of traditional convolution 2D layers is limited. This helps to accelerate the processing speed in the real scenario, but at the cost of accuaracy.

## Model

### 1. The model is submitted in the correct format.

see `model_weights.h5` and `config_model_weights.h5`

### 2. The neural network must achieve a minimum level of accuracy for the network implemented.

weight = 0.7639821029082774

```
final_Iou = 0.581808046281

final_score = 0.444490934687
```