Project Follow me

1. Writeup
   1.1. Criteria 1.1

Criteria	Meets Specifications	Result
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.	The write-up / README should include a statement and supporting figures / images that explain how each rubric item was addressed, and specifically where in the code each step was handled. The write-up should include a discussion of what worked, what didn't and how the project implementation could be improved going forward. This report should be written with a technical emphasis (i.e. concrete, supporting information and no 'hand-waving'). Specifications are met if a reader would be able to replicate what you have done based on what was submitted in the report. This means all network architecture should be explained, parameters should be explicitly stated with factual justifications, and plots / graphs are used where possible to further enhance understanding. A discussion on potential improvements to the project submission should also be included for future enhancements to the network / parameters that could be used to increase accuracy, efficiency, etc. It is not required to make such enhancements, but these enhancements should be explicitly stated in its own section titled "Future Enhancements".	**Specification met** Write Up including images and figures explaining the architecture is provided The table describing hyper parameter tuning in criteria 1.3 explains what steps worked and didn't work to achieve the final score IOU > 0.4 A section related to future enhancements is available at the end

   1.2. Criteria 1.2

Criteria	Meets Specifications	Result
The write-up conveys the an understanding of the network architecture.	The student clearly explains each layer of the network architecture and the role that it plays in the overall network. The student can demonstrate the benefits and/or drawbacks different network architectures pertaining to this project and can justify the current network with factual data. Any choice of configurable parameters should also be explained in the network architecture. The student shall also provide a graph, table,	**Specification met.** The network architecture is shown below with a figure. The reason for choice of a FCN is described. The configurable parameters are described.

	diagram, illustration or figure for the overall network to serve as a reference for the reviewer.	

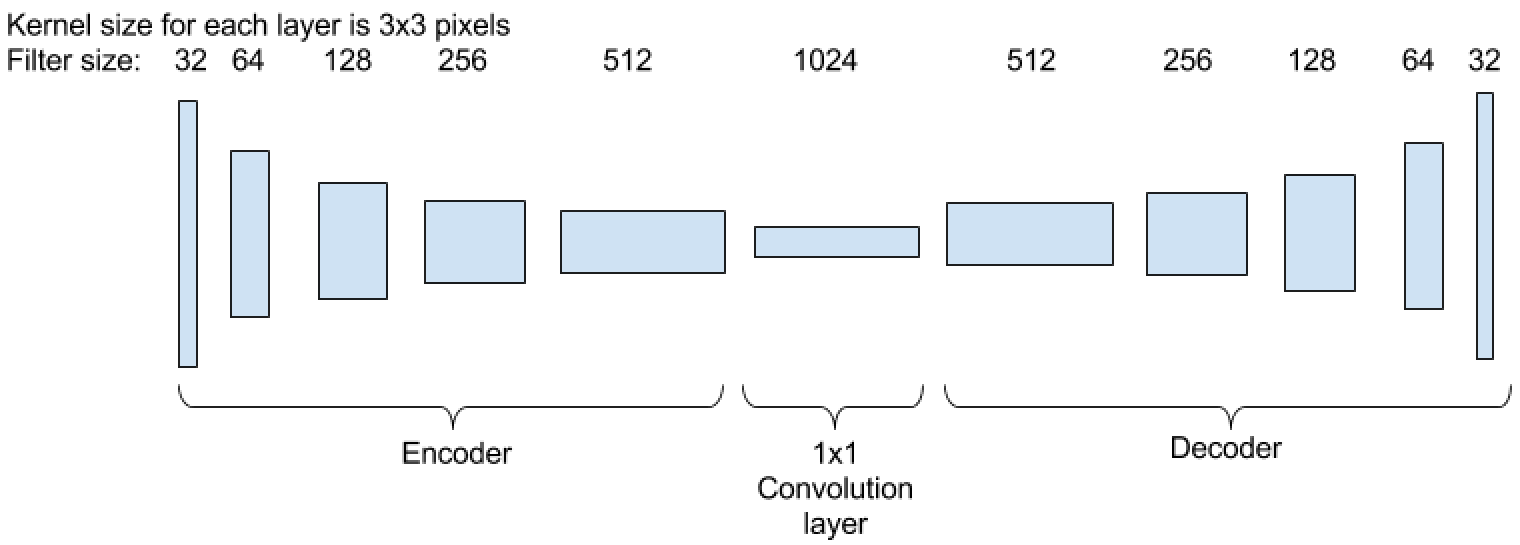
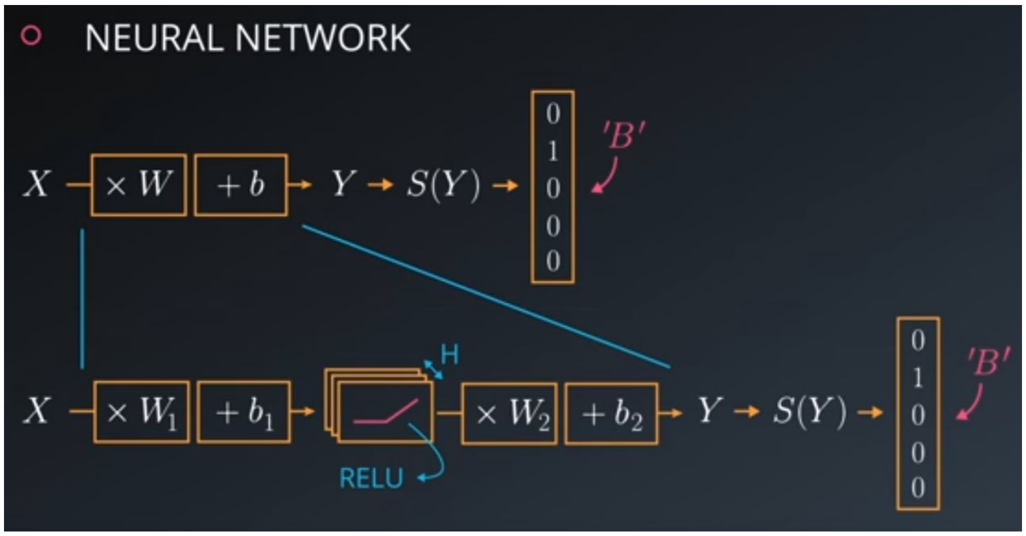

Kernel size for each layer is 3x3 pixels
Filter size: 32 64 128 256 512 1024 512 256 128 64 32
Encoder
1x1 Convolution layer
Decoder

Fig. Architecture of the FCN used in the project

- A FCN - Fully convolutional network is used to identify the hero in the crowd. A CNN can tell us if the hero is present in the image. To achieve follow me function, it is necessary to identify the location of hero within the image. FCN helps us achieve semantic segmentation. The network comprises of a encoder section, a 1x1 convolution layer and a decoder layer. Note: The number of filters in each layer was tuned to achieve the required performance of IOU > 0.4. The steps taken are listed in the table in criteria 1.3
- In a typical Convnet, we use triads to reduce the dimensionality in each layer and increase the depth layer after layer. After achieving the necessary depth, the output is connected to a set of fully connected layers. In a fully convolutional network, the fully connected layers are replaced by a 1x1 convolution layer and then upsampled by transposed convolution layers. In this case a bilinear upsampling method is used. The encoder tells us if the hero is present in the image while the decoder section tells us where in the image the hero is present.
- Each convolutional layer will comprise of a set of weights, followed by a bias. This is based on the linear equation y = WX + B where W are the weights, X the inputs and B the bias. Following this, a RELU unit is added to incorporate nonlinearity in the neural network.



- The RELU unit is followed by a max pool function. Striding is a very aggressive way of down sampling. Pooling allows to take all the convolutions in the neighbourhood and combines them by averaging , taking the maximum value etc. Here max pooling is used. Max pooling is commonly used because it doesn't add parameters and doesn't risk the increase of overfitting while making the layer more accurate. The downside is that the computation cost is higher because the stride value is now low.

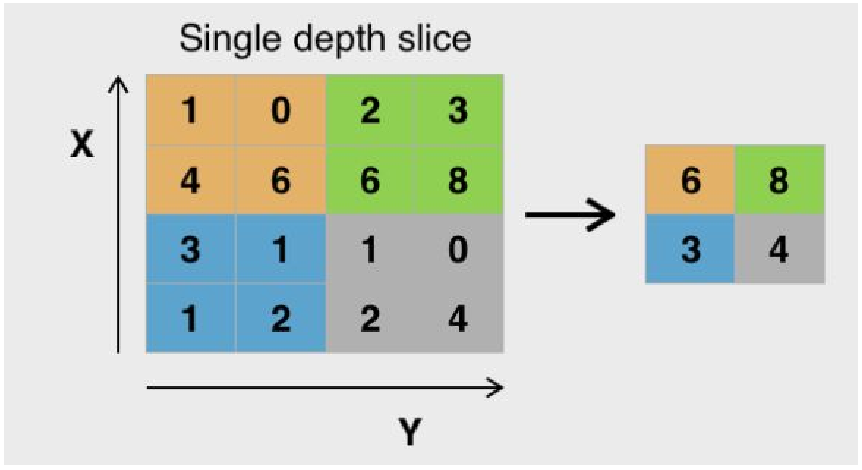


Fig. Max pooling

- Dropout layers are added between each convolutional layer - This is a regularization technique where half of the activations between each layer is randomly set to zero This forces the network to learn a redundant representation of everything to make sure at least some of the information remains. The dropout probability is set to 0.5 typically.
- Keras abstracts the structure of the underlying convolutional network. The code is provided below.

```
def separable_conv2d_batchnorm(input_layer, filters, strides=1):
    output_layer = SeparableConv2DKeras(filters=filters,kernel_size=3,
    strides=strides,padding='same', activation='relu')(input_layer)

    output_layer = layers.Batch Normalization()(output_layer)
    return output_layer
```

  - **Parameters:**
  - <Filters> is the number of filters to be applied. This is determined by the number of unique features you would like to capture. Example, for a dog, a separate filter is required to capture the teeth, tongue etc.
  - <Kernel_size> is the size of the patch in pixels. In this case it is a 3x3 pixel patch.
  - <strides> is the number of pixels that gets shifted each time the kernel is swept across the image
  - <padding> - Two types of padding are possible viz., same and valid padding. Same padding is used where the output image remains the same same size of the input by padding the input with zeroes.
  - <activation> - Here RELU is used to incorporate nonlinearity into the model

- A softmax function at the end helps to calculate the probabilities of the different classes. In this case, it determines the probability of the hero present in the image.

```
return layers.Conv2D(num_classes, 1, activation='softmax',
padding='same')(dec_conv_layer5)
```

- Other configurable parameters used in the CNN lab are listed below. These parameters are abstracted by the Keras framework which is used to build the network in the Project.
  - Keep_probabiliy: Set keep_probability to the probability of keeping a node using dropout

    - weight = tf.Variable(tf.truncated_normal(filter_size, stddev = 0.01)) . The stddev gives the standard deviation of the randomly selected weights.

- Encoder block implementation (single layer):

```
def encoder_block(input_layer, filters, strides):

    # TODO Create a separable convolution layer using the separable_conv2d_batchnorm()
function.
    output_layer = separable_conv2d_batchnorm(input_layer,filters,strides)

    return output_layer
```

- Decoder block implementation (single layer):

```
def decoder_block(small_ip_layer, large_ip_layer, filters):

    # TODO Upsample the small input layer using the bilinear_upsample() function.
    upsampled_layer = bilinear_upsample(small_ip_layer)

    # TODO Concatenate the upsampled and large input layers using layers.concatenate
    concatenated_output = layers.concatenate([upsampled_layer, large_ip_layer])

    # TODO Add some number of separable convolution layers
    output_layer = separable_conv2d_batchnorm(concatenated_output, filters, strides=1)
    return output_layer
```

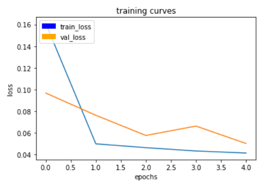
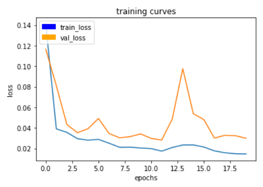
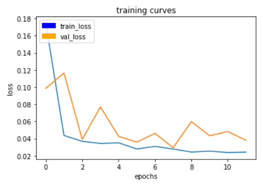
- Fully assembled network:

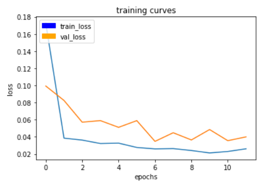
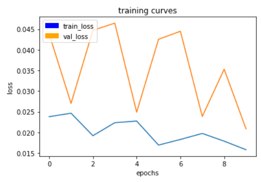
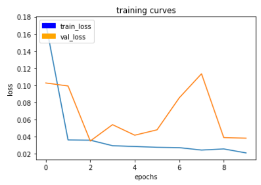
```
def fcn_model(inputs, num_classes):
   # TODO Add Encoder Blocks.
    # Remember that with each encoder layer, the depth of your model (the number of
filters) increases.
    enc_conv_layer1 = encoder_block(inputs, 32, 2)
    enc_conv_layer2 = encoder_block(enc_conv_layer1, 64, 2)
    enc_conv_layer3 = encoder_block(enc_conv_layer2, 128, 2)
    enc_conv_layer4 = encoder_block(enc_conv_layer3, 256, 2)
    enc_conv_layer5 = encoder_block(enc_conv_layer4, 512, 2)



    # TODO Add 1x1 Convolution layer using conv2d_batchnorm().
    conv_1x1 = conv2d_batchnorm(enc_conv_layer5, 512, kernel_size=1, strides=1)

    # TODO: Add the same number of Decoder Blocks as the number of Encoder Blocks
    dec_conv_layer1 = decoder_block(conv_1x1, enc_conv_layer4, 256)
    dec_conv_layer2 = decoder_block(dec_conv_layer1, enc_conv_layer3, 128)
    dec_conv_layer3 = decoder_block(dec_conv_layer2, enc_conv_layer2, 64)
    dec_conv_layer4 = decoder_block(dec_conv_layer3, enc_conv_layer1, 32)
    dec_conv_layer5 = decoder_block(dec_conv_layer4, inputs, 16)

    # The function returns the output layer of your model. "x" is the final layer obtained
from the last decoder_block()
    return layers.Conv2D(num_classes, 1, activation='softmax',
padding='same')(dec_conv_layer5)
```

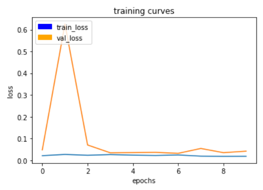
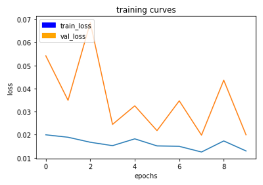
1.3. Criteria 1.3

Criteria	Meets Specifications	Result
The write-up conveys the student's understanding of the parameters chosen for the the neural network.	The student explains their neural network parameters including the values selected and how these values were obtained (i.e. how was hyper tuning performed? Brute force, etc.) Hyper parameters include, but are not limited to: Epoch Learning Rate Batch Size Etc. All configurable parameters should be explicitly stated and justified.	Result : Specification met The hyper tuning parameters and the steps taken during tuning are described below.

1. The hyper parameters are
    a. **Learning rate:** This value gets multiplied to the weights every time the parameters are tuned to move the line that separates the classes. A large value will make large movements and can cause multiple wrong classifications. Hence a smaller value is preferable. A smaller value on the other hand increases the computation cost.
    b. **batch_size**: number of training samples/images that get propagated through the network in a single pass.
    c. **num_epochs**: number of times the entire training dataset gets propagated through the network.
    d. **steps_per_epoch**: number of batches of training images that go through the network in 1 epoch. We have provided you with a default value. One recommended value to try would be based on the total number of images in training dataset divided by the batch_size.
    e. **validation_steps**: number of batches of validation images that go through the network in 1 epoch. This is similar to steps_per_epoch, except validation_steps is for the validation dataset. We have provided you with a default value for this as well.
    f. **workers**: maximum number of processes to spin up. This can affect your training speed and is dependent on your hardware. We have provided a recommended value to work with.
2. To tune the hyper parameters two major steps were performed
    a. Create a baseline performance level. We select a set of hyper parameters and run it on the neural network of the Segmentation lab exercise. These hyper parameters selection were guided by the hyper parameter tuning results from the CNN lab exercise. The iterations are described in the table below.
    b. We start with the hyper parameters values and then iteratively improve the performance of the network by increasing the layers of the network and by changing the hyper parameters. The iterations are described in the table below.

Hyperparameters	Epoch final	Follow target	patrol/target not visible	Detect target from farway	Final score
Segmentation lab - Training					
Run 1: Initial set selected at random. Two convolutions in the encoder followed by 1x1 convolution layer and two convolutions in the decoder. The Filter sizes for the two convolutions were 16 and 32. Filter size of the 1x1 convolution layer was 64.					
learning_rate = 0.01 batch_size = 32 num_epochs = 5 steps_per_epoch = 200 validation_steps = 50 workers = 2	 200/200 [=============================] - 51s - loss: 0.0414 - val_loss: 0.0502	number of validation samples intersection over the union evaluated on 542 average intersection over union for background is 0.9860105508605038 average intersection over union for other people is 0.19654127934724738 average intersection over union for the hero is 0.571284747548524 number true positives: 539, number false positives: 0, number false negatives: 0	number of validation samples intersection over the union evaluated on 270 average intersection over union for background is 0.9674994072859483 average intersection over union for other people is 0.3046649013921299 average intersection over union for the hero is 0.0 number true positives: 0, number false positives: 97, number false negatives: 0	number of validation samples intersection over the union evaulated on 322 average intersection over union for background is 0.9937493581663888 average intersection over union for other people is 0.29812961115206743 average intersection over union for the hero is 0.1168703983883113 number true positives: 109, number false positives: 4, number false negatives: 192	0.236941835583
Run 2: Increased the number of epochs from 5 to 20. The final score increased					
learning_rate = 0.01 batch_size = 32 num_epochs = 20 steps_per_epoch = 200 validation_steps = 50 workers = 4	 200/200 [==============================] - 60s - loss: 0.0146 - val_loss: 0.0299	number of validation samples intersection over the union evaulated on 542 average intersection over union for background is 0.9954956821989862 average intersection over union for other people is 0.2975085806314347 average intersection over union for the hero is 0.8741515799227858 number true positives: 539, number false positives: 0, number false negatives: 0	number of validation samples intersection over the union evaulated on 270 average intersection over union for background is 0.9872567281094876 average intersection over union for other people is 0.7339875440653135 average intersection over union for the hero is 0.0 number true positives: 0, number false positives: 18, number false negatives: 0	number of validation samples intersection over the union evaulated on 322 average intersection over union for background is 0.9963916227699875 average intersection over union for other people is 0.3680376956755425 average intersection over union for the hero is 0.06905469156238567 number true positives: 64, number false positives: 0, number false negatives: 237	0.33144136463
Run 3: Experimented with halving the learning rate. The scores did not improve. This could be because of the limited data used to train the network. Some improvement may have been achieved by increasing the steps_per_epoch and by reducing the batch size. These steps were not done and this value was taken as the baseline for the Project follow me.					
learning_rate = 0.005 batch_size = 16 num_epochs = 12 steps_per_epoch = 200 validation_steps = 50 workers = 4	 200/200 [=============================] - 49s - loss: 0.0245 - val_loss: 0.0380	number of validation samples intersection over the union evaulated on 542 average intersection over union for background is 0.9954956821989862 average intersection over union for other people is 0.2975085806314347 average intersection over union for the hero is 0.8741515799227858 number true positives: 539, number false positives: 0, number false negatives: 0	number of validation samples intersection over the union evaulated on 270 average intersection over union for background is 0.9872567281094876 average intersection over union for other people is 0.7339875440653135 average intersection over union for the hero is 0.0 number true positives: 0, number false positives: 18, number false negatives: 0	number of validation samples intersection over the union evaulated on 322 average intersection over union for background is 0.9963916227699875 average intersection over union for other people is 0.3680376956755425 average intersection over union for the hero is 0.06905469156238567 number true positives: 64, number false positives: 0, number false negatives: 237	0.33144136463

Project follow me - Training					
Run 1: Baseline run using the same hyper parameters and network architecture from the segmentation lab.					
learning_rate = 0.005 batch_size = 16 num_epochs = 12 steps_per_epoch = 200 validation_steps = 50 workers = 4	 200/200 [==============================] - 49s - loss: 0.0261 - val_loss: 0.0400	number of validation samples intersection over the union evaulated on 542 average intersection over union for background is 0.9819799551483401 average intersection over union for other people is 0.27995951117211443 average intersection over union for the hero is 0.6945138066319383 number true positives: 539, number false positives: 0, number false negatives: 0	number of validation samples intersection over the union evaulated on 270 average intersection over union for background is 0.982054585465969 average intersection over union for other people is 0.6029510311909622 average intersection over union for the hero is 0.0 number true positives: 0, number false positives: 121, number false negatives: 0	number of validation samples intersection over the union evaulated on 322 average intersection over union for background is 0.9943681316988896 average intersection over union for other people is 0.3364969780610734 average intersection over union for the hero is 0.10846548918865527 number true positives: 91, number false positives: 1, number false negatives: 210	0.26 292 981 100 2
Run 2: Increase depth of the neural network. The number of layers in the encoder was increased from 2 to 5. The value 5 was a random selection and was not optimized for computation cost or for classification performance. Encoder layer filter sizes: 16, 32,64, 128, 256. 1x1 convolution layer size: 512. Increased the workers to 4 since the aws instance is a quad core machine. Increase epochs to 20 Increase workers = 4 Encoders = 5 layers, Decoders = 5 layers					
learning_rate = 0.005 batch_size = 16 num_epochs = 10 steps_per_epoch = 200 validation_steps = 50 workers = 4	 200/200 [==============================] - 49s - loss: 0.0158 - val_loss: 0.0208	number of validation samples intersection over the union evaulated on 542 average intersection over union for background is 0.9952299264390188 average intersection over union for other people is 0.29305081085811896 average intersection over union for the hero is 0.8906767395717298 number true positives: 539, number false positives: 0, number false negatives: 0	number of validation samples intersection over the union evaulated on 270 average intersection over union for background is 0.986189876568267 average intersection over union for other people is 0.7061475226633541 average intersection over union for the hero is 0.0 number true positives: 0, number false positives: 45, number false negatives: 0	number of validation samples intersection over the union evaulated on 322 average intersection over union for background is 0.9961941095891161 average intersection over union for other people is 0.3772516893923592 average intersection over union for the hero is 0.124994537269569 number true positives: 96, number false positives: 1, number false negatives: 205	0.36 396 798 013 2
Run 3: Increase the number of filters x2. Filter size: 32, 64, 128, 256, 512. 1x1 convolution layer size: 1024. In run 2, after epoch =12 , training and value loss start to increase. Reduce batch size to 16. Reduce the learning rate to 0.005					
learning_rate = 0.005 batch_size = 16 num_epochs = 10 steps_per_epoch = 200 validation_steps = 50 workers = 4		number of validation samples intersection over the union evaulated on 542 average intersection over union for background is 0.9937017132383823 average intersection over union for other people is 0.25486026319770005 average intersection over union for the hero is 0.8798185784496415 number true positives: 538, number false positives: 0, number false negatives: 1	number of validation samples intersection over the union evaulated on 270 average intersection over union for background is 0.9805624439113946 average intersection over union for other people is 0.6010921137128773 average intersection over union for the hero is 0.0 number true positives: 0, number false positives: 40, number false negatives: 0	number of validation samples intersection over the union evaulated on 322 average intersection over union for background is 0.9949051243087009 average intersection over union for other people is 0.33522217743083627 average intersection over union for the hero is 0.15561131636825554 number true positives: 107, number false positives: 1, number false negatives: 194	0.37 903 080 712 7

Run 4: Reduced steps per epoch to 100 and found the score to worsen					
learning_rate = 0.005 batch_size = 16 num_epochs = 10 steps_per_epoch = 100 validation_steps = 50 workers = 4		number of validation samples intersection over the union evaulated on 542 average intersection over union for background is 0.9925153446616675 average intersection over union for other people is 0.24947427677297498 average intersection over union for the hero is 0.738228211583109 number true positives: 499, number false positives: 0, number false negatives: 40	number of validation samples intersection over the union evaulated on 270 average intersection over union for background is 0.9818507806193012 average intersection over union for other people is 0.7064051370063723 average intersection over union for the hero is 0.0 number true positives: 0, number false positives: 2, number false negatives: 0	number of validation samples intersection over the union evaulated on 322 average intersection over union for background is 0.9917537802789214 average intersection over union for other people is 0.3217546151480016 average intersection over union for the hero is 0.08022099286462943 number true positives: 54, number false positives: 0, number false negatives: 247	0.268766276757
Run 5: Increased the steps per epoch to 400 and achieved the necessary score of > 0.4					
learning_rate = 0.005 batch_size = 16 num_epochs = 10 steps_per_epoch = 400 validation_steps = 50 workers = 4	 400/400 [=============================] - 94s - loss: 0.0130 - val_loss: 0.0200	number of validation samples intersection over the union evaulated on 542 average intersection over union for background is 0.9959552741389014 average intersection over union for other people is 0.35275427986438035 average intersection over union for the hero is 0.9019598431582083 number true positives: 539, number false positives: 0, number false negatives: 0	number of validation samples intersection over the union evaulated on 270 average intersection over union for background is 0.9896984136683329 average intersection over union for other people is 0.7913892817508856 average intersection over union for the hero is 0.0 number true positives: 0, number false positives: 33, number false negatives: 0	number of validation samples intersection over the union evaulated on 322 average intersection over union for background is 0.9969325910153842 average intersection over union for other people is 0.44974463952743726 average intersection over union for the hero is 0.2096986714053058 number true positives: 122, number false positives: 1, number false negatives: 179	0.420369724329

1.4. Criteria 1.4

Criteria	Meets Specifications	Result
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.	The student is demonstrates a clear understanding of 1 by 1 convolutions and where/when/how it should be used. The student demonstrates a clear understanding of a fully connected layer and where/when/how it should be used.	Result: Specification met The use and implementation of fully connected layer and 1x1 convolution is described below.

1x1 Convolution layer:

- A 1x1 convolution looks at only one pixel and not a patch of pixels.
- Interspersing the convolution layers with a 1x1 convolution layer makes the network deeper and gives us more parameters without increasing the computation cost. They are less expensive computationally because they are not convolutions but only matrix multipliers. Note: It is better to have a deeper network than a wider network to achieve better classification.
- In a fully convolutional network, instead of a fully connected layer a 1x1 convolution layer is used. This helps to avoid flattening the output of the encoder to 2D and retains the 4D information. 4D information allows the layer to be upsampled by the decoder to retrieve the original image.

- It is implemented by specifying Kernel and stride as 1

  ```
  Example:
  conv_1x1 = conv2d_batchnorm(enc_conv_layer5, 512, kernel_size=1, strides=1),
  which in turn calls,
  output_layer = layers.Conv2D(filters=filters, kernel_size=kernel_size,
  strides=strides, padding='same', activation='relu')(input_layer)

   output_layer = layers.Batch Normalization()(output_layer)
  ```

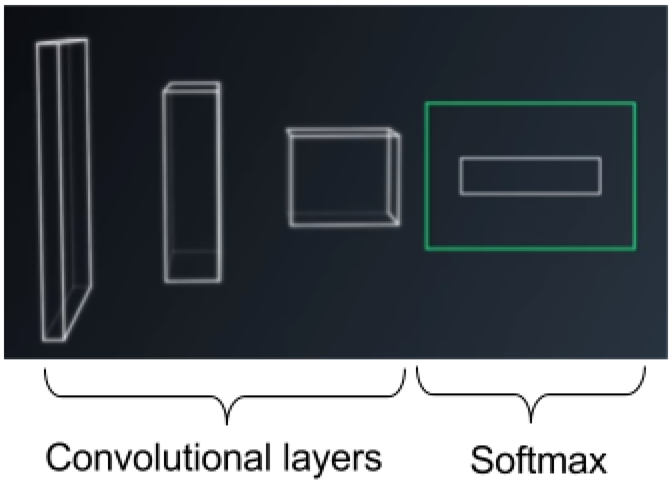
Fully connected layers:

- The output from the convolutional layers represents high-level features in the data. While that output could be flattened and connected to the output layer, adding a fully-connected layer is a cheap way of learning non-linear combinations of these features.
- Finally, after several convolutional and max pooling layers, the high-level reasoning in the neural network is done via fully connected layers. Neurons in a fully connected layer have connections to all activations in the previous layer, as seen in regular neural networks. Their activations can hence be computed with a matrix multiplication followed by a bias offset.
- It is implemented by using tf.layers.dense

```
Example: fc_layer = tf.layers.dense(x_tensor, num_outputs, activation=tf.nn.relu)
```

1.5. Criteria 1.5

Criteria	Meets Specifications	Result
The student has a clear understanding of image manipulation in the context of the project indicated by the write-up.	The student is able to identify the use of various reasons for encoding / decoding images, when it should be used, why it is useful, and any problems that may arise.	Result: Specification met The use of encoder and decoder layer is described. The problem faced during upsampling and the mitigation technique is described.

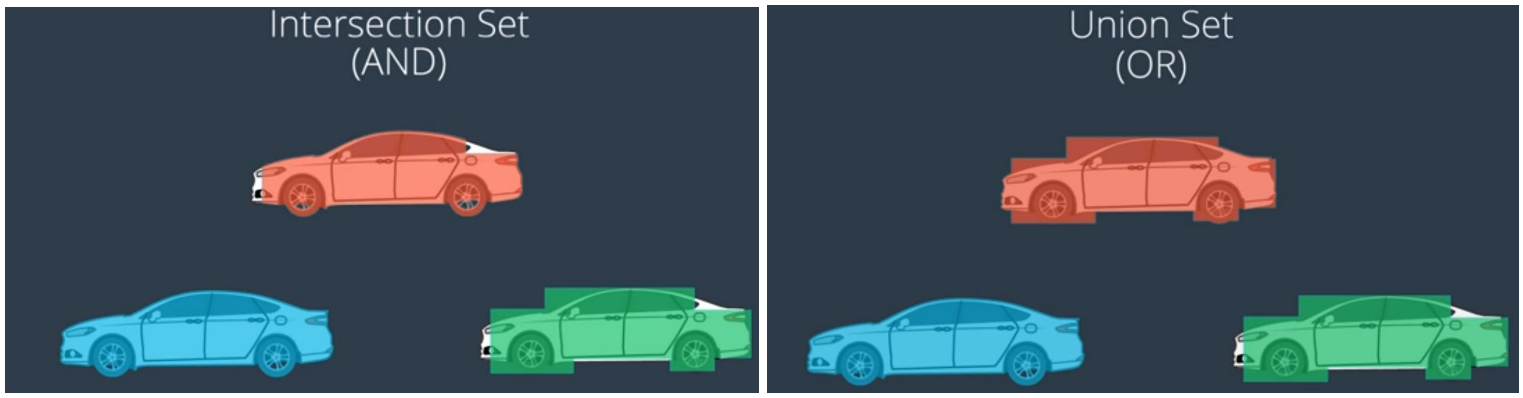


- A typical convolutional network comprises of convolutions followed by a softmax function which gives the probability which in turn tells us if the desired subject is in the picture or not.

- Each convolutional layer progressively down samples the image and removes the spatial content while leaving the semantic complexity of the image. Hence the network can tell us whether the desired subject is in the image but it cannot tell us where in the image the subject is located.
- For example, applications where a robot needs to identify an object and then calculate the pose estimate of the object, it is necessary to know where in the image the classified object resides. Its also useful in mobile robot applications where the drivable surface needs to be identified.
- During the upsampling, a lot of the details from the original image may be missed out. This may lead to wrong semantic classification. For example: Non - drivable surface of the road may be classified as drivable and vice versa. To avoid this, skipped connections and layer concatenations become useful. These connections directly connect the image from specific layers of the encoder and add or concatenate them to the decoder layers. This way the details of the image are retained. This is implemented in the code as:

```
concatenated_output = layers.concatenate([upsampled_layer, large_ip_layer])
```

  This function is from tensorflow.contrib.keras.python.keras import layers
- The Intersection over Union metric helps us determine how well we have identified the contours of the subject within the image.
  IOU = Intersection set / union set



- IOU is implemented in tensor through the tf.metrics.mean_iou function. This function, returns a Tensor for the metric result and a Tensor Operation to generate the result. In this case it returns mean_iou for the result and update_op for the update operation.

```
sess.run(update_op)
sess.run(mean_iou)
```

1.6. Criteria 1.6

Criteria	Meets Specifications	Result
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.	The student is able to clearly articulate whether this model and data would work well for following another object (dog, cat, car, etc.) instead of a human and if not, what changes would be required.	Result: Specification met The process behind training a network for a specific subject is described below.

- The same model trained with the hero images cannot identify other subjects because it is not trained to identify other images. With enough data and training, the network may be able to classify the desired subject but the performance of the network determined by the IOU metric may not meet the desired performance expectation.
- The lack of performance can be attributed to mainly the architecture of the network and the hyper parameters tuning.
- The number of features that make up the subject and the resolution of the training set will all determine whether the network is deep enough to identify the subject. For example: A dog is characterized by a red tongue, teeth and a nose. Based on the image resolution, if these features are dimensionally very small compared to the overall image, a deeper network will be required. The number of filters will be determined by the number of unique features we want to capture. Hence a network to identify a car may not be able to identify a dog in images of similar resolution.
- Hyper parameters such as the learning rate, batch size, steps per epoch all determine how efficiently the network trains on the images. A more complex subject will require a lower learning rate and a batch size. If the learning rate is unable to give us the desired performance, the network would have to be deeper.

2. Model

2.1. Criteria 2.1

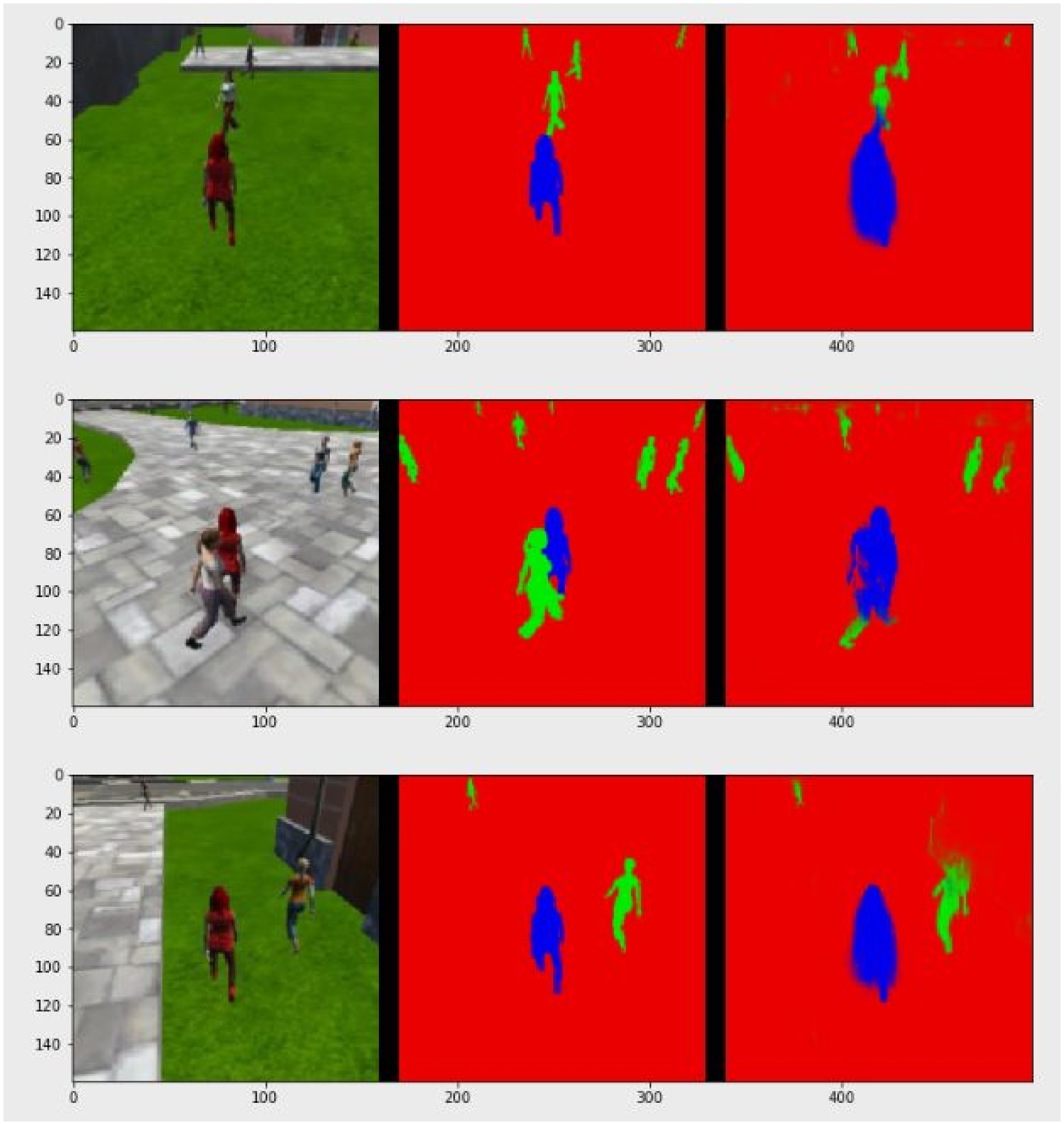
Criteria	Meets Specifications	Result
The model is submitted in the correct format.	The file is in the correct format (.h5) and runs without errors.	Result: Specification met Attached: model_weights.h5

2.2. Criteria 2.2

Criteria	Meets Specifications	Result
The neural network must achieve a minimum level of accuracy for the network implemented.	The neural network should obtain an accuracy greater than or equal to 40% (0.40) using the Intersection over Union (IoU) metric.	Result: Specification met Final score: 0.420369724329 (42.03%)

Future enhancement:

- The network depth was picked randomly and the required score was achieved by basic hyper parameter tuning. There is scope to reduce the depth of the network and realize a higher score through hyper parameter tuning. The four main areas to focus are : number of epochs, learning rate, batch size, steps per epoch.
- The filter size was at best chosen at random followed by some iterations during the hyper parameter turning. The main objective was to achieve the IOU score and not enough emphasis was given to getting the features detected correctly. An example is shown below. This can be rectified by tuning the filter size and the network depth. It can be noted that the third column boxes which show the detected hero in frame do not exactly match the contours of the hero. I need more inputs on how to select the size of the patch to ensure that the features are captured.



- The final score of 0.42 is quite low to achieve a stable lock on the hero. IOU score has to be improved. I need inputs on how to set the IOU target for a project.