Follow Me Project Udacity RoboND

Writeup report.
For information on how to run project visit README

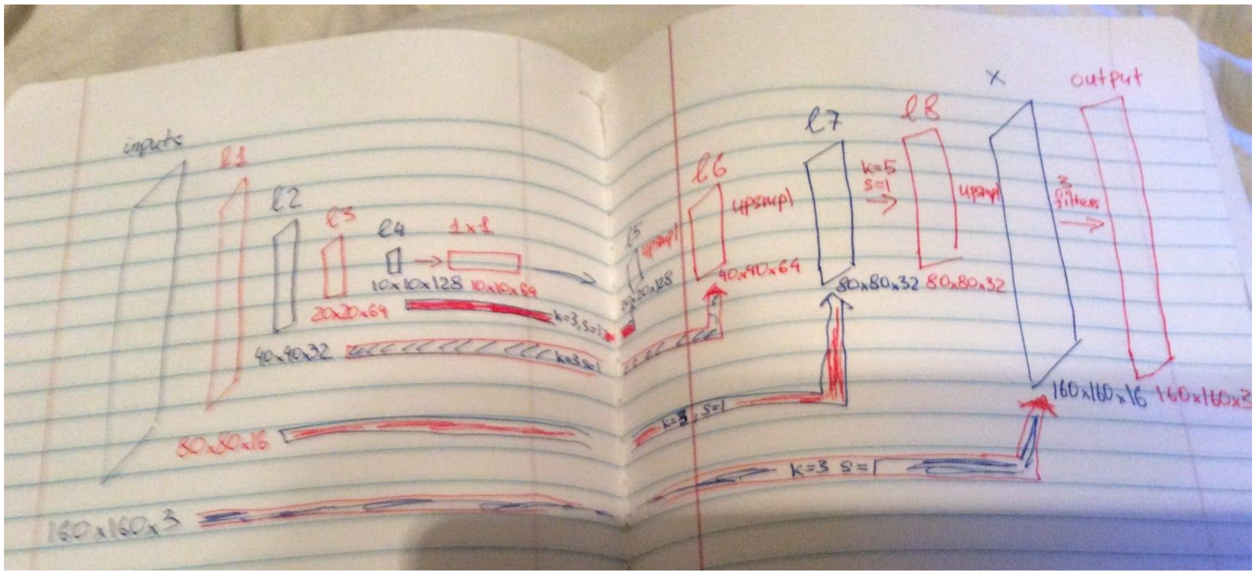
Additional data

Additional 600 images were added to the training set. These images were taken following the hero manually and most of them contain hero either close-up or in the distance

Network Architecture

**Deep Network Model **

The network consists of 4 encoder and 4 decoder layers (default kernel_size=7) with additional 1x1 convolution in the middle as well as a convolution with (kernel_size=5, strides = 1) before the final layer. As you can see he 1x1 convolution actually decreases the amount of filters (depth) of the layer.



The model was run with 3 encoder/decoder layers first and gave final score of .37. After adding additional layer the score the while leaving filter size of 1x1 convolution the same increased the final score by .04 and helped achieve passing submission. Run larger amount of filters (256) on 1x1 convolution and the result passed the submission criteria, but was slightly lower than with 64 filters..

** Encoder Layer **

Consists of a separable convolution with kernel_size=7 and strides=2 decreasing the image size by 2x2
Each consecutive encoder layer doubles the amount of filters applied - 16 for layer1 to 128 for layer4 resulting in increased feature map for each downsampled pixel and higher depth of the layer
TODO: might benefit from more layers

** 1x1 convolutions **

1x1 convolution is used between encoder and decoder layers. The convolution of kernel size 1 and strides 1 decreases the depth of the layer from 128 to 64, but preserves the size and results in a matrix multiplication. It is then used by the decoder to upsample the image back to original size and preserve spatial information from the image. Makes it possible to use different sizes of images to train the network.

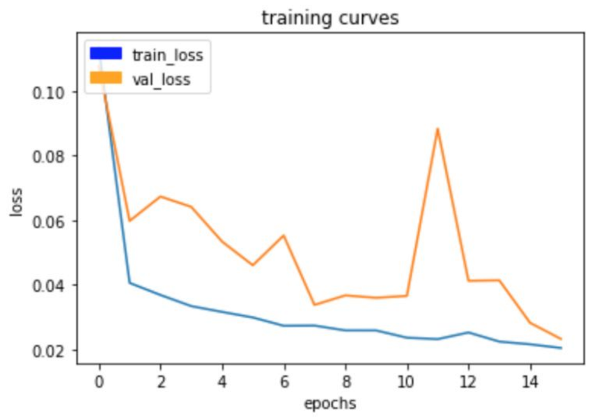
** Decoder Layer **

Takes working layer and a skipped connection layer that is 2x2 larger
Performs bilinear upsampling on the working layer and concatenates it with skipped one

After that performs separable convolution on a skipped layer with (kernel_size=3, strides=1) with no downsampling
Concatenates these two layers.
Resulting output_layer is 2x2 larger than the working layer and the same size as pass through layer.
TODO: not sure if it is doing something great over there, but computing time increases and result is slightly better

** Hyperparameters **

Learning Rate: 0.01 - Higher learning rate causes unexpected val_loss increases, while lower ones make the model improve loss too slowly
Batch size: 32 - with p2.xlarge instances the amount of layers causes the system to occasionally run out of resources with batch size 64.
Num epochs: 20 - I figured out that large amount of epochs make network overfit to the training data spiking (and possibly validation data) the val loss and the final result. The weights used for the model - epoch 16 weights with lowest val_loss



```
150/149 [==============================] - 89s - loss: 0.0205 - acc: 0.9937 - val_loss: 0.023
3 - val_acc: 0.9936
```

Steps per epoch: 149 - amount of training data/batch size

** Changes in the model **

1. Change in optimizer to Nadam which is better with my learning rate.
2. Add ModelCheckpoint callback to record weights with best val_loss

## Result

After experimenting with hyperparameters and layers the following setup produced needed final_result of 0.41
The network configuration is saved at: /data/weights/amazing_aws_weights.h5
To run the network with these weights load from model_training.ipynb

```
In [68]: # And the final grade score is
         final_score = final_IoU * weight
         print(final_score)

         0.412978302912
```

```
# Scores for while the quad is following behind the target.
true_pos1, false_pos1, false_neg1, iou1 = scoring_utils.score_run_iou(val_following, pred_foll
owing)
```

```
number of validation samples intersection over the union evaulated on 542
average intersection over union for background is 0.9945373870382439
average intersection over union for other people is 0.23393614845789387
average intersection over union for the hero is 0.8438514220752148
number true positives: 537, number false positives: 0, number false negatives: 2
```

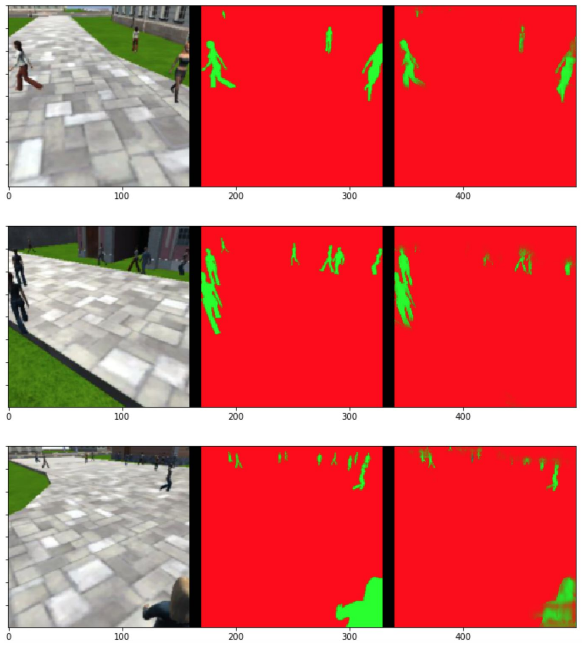
```
# Scores for images while the quad is on patrol and the target is not visable
true_pos2, false_pos2, false_neg2, iou2 = scoring_utils.score_run_iou(val_no_targ, pred_no_tar
g)
```

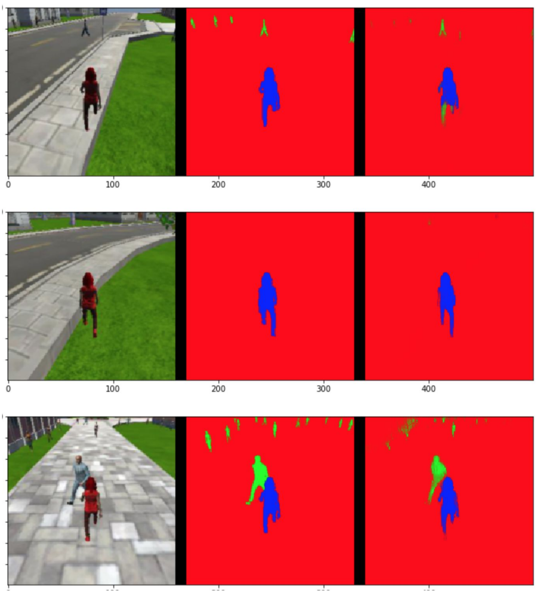
```
number of validation samples intersection over the union evaulated on 270
average intersection over union for background is 0.9821361131288673
average intersection over union for other people is 0.6106455650037486
average intersection over union for the hero is 0.0
number true positives: 0, number false positives: 26, number false negatives: 0
```

```
# This score measures how well the neural network can detect the target from far away
true_pos3, false_pos3, false_neg3, iou3 = scoring_utils.score_run_iou(val_with_targ, pred_with
_targ)
```

```
number of validation samples intersection over the union evaulated on 322
average intersection over union for background is 0.9956149035633401
average intersection over union for other people is 0.3313734980351461
average intersection over union for the hero is 0.22142396255459967
number true positives: 136, number false positives: 2, number false negatives: 165
```

The results from higher distance are lower which might attribute to low amount of filters in 1x1 convolution. The network performed well in following the hero and patrolling noticing other people.





** Changing training classes **

A change in training class will require to tune in layers and hyperparameters. Smaller objects may need a higher downsampling/upsampling size than the hero. The training data should focus more on the object needed to be recognize which is hero in the project above. Since the network is recognizing masked images any pictures with appropriate masks should get the model trained if they are sufficient in the training and validation set.