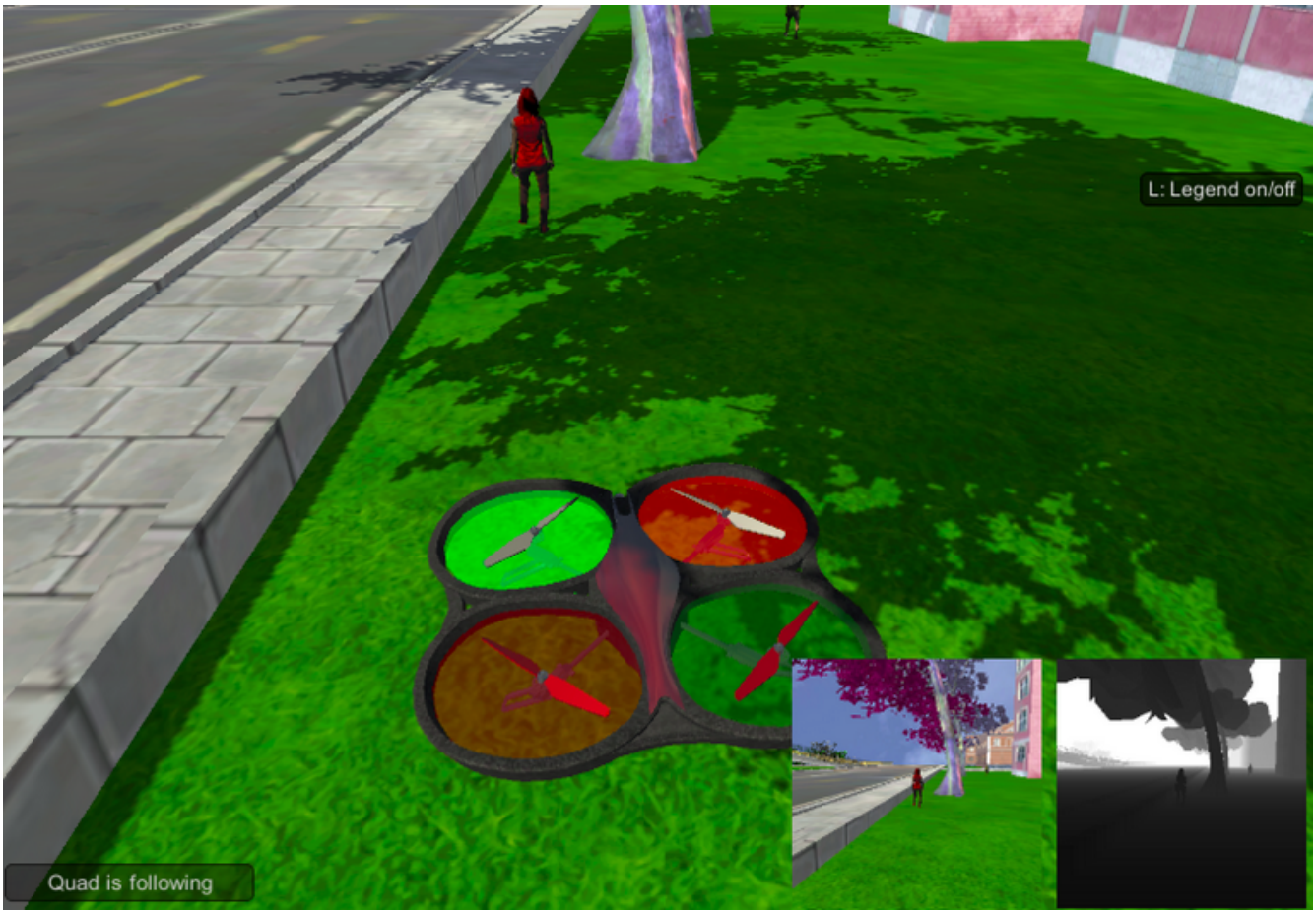
Arturo Polanco Lozano
Self Driving Car – Robotics Engineer
Report Follow Me Project

Introduction

This project uses a fully convolutional neural network to let a drone segment a target in an image and follow it in a virtual environment. This aproach allows to paint every single pixel in an image depending on the label of the object it contains and in order to measure the performance of the model is used IOU (intersection over union) which takes the intersection of the prediction pixels and ground trurh pixels and devides it by the union of them.



Data Collection

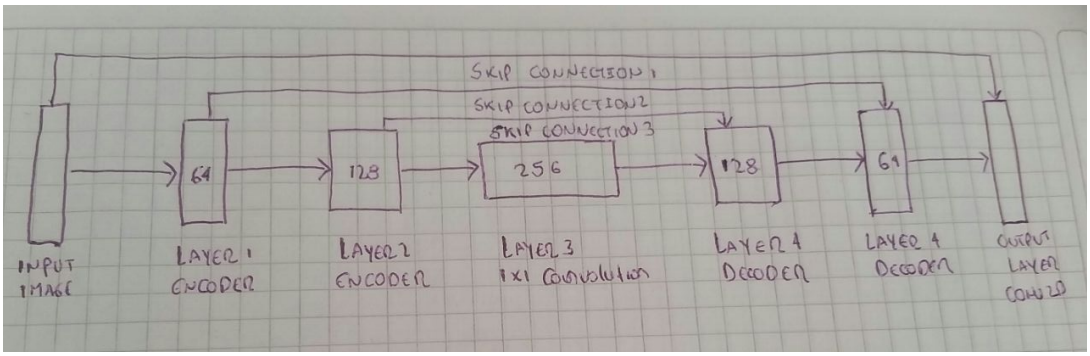
I used the data provided by Udacity.

Network Architecture

The network can be devided into the following stages:

- Input image with a skip connection with the output convolutional layer, skip connections are important due to they allow the network to use information from multiple resolution scales and helps to make more precise segmentation decisions.
- Firs encoder layer, which extracts characteristics useful to perform segmentation.

- Second encoder layer which has a skip connection with the first decoder.
- 1x1 Convolution layer, this is an alternative of a fully connected with the advantage that preserves spatial information.
- First decoder layer, that up scale the encoder output back into the dimensions of the original image.
- Second decoder layer.
- Convolution output with softmax activation function to perform segmentation.



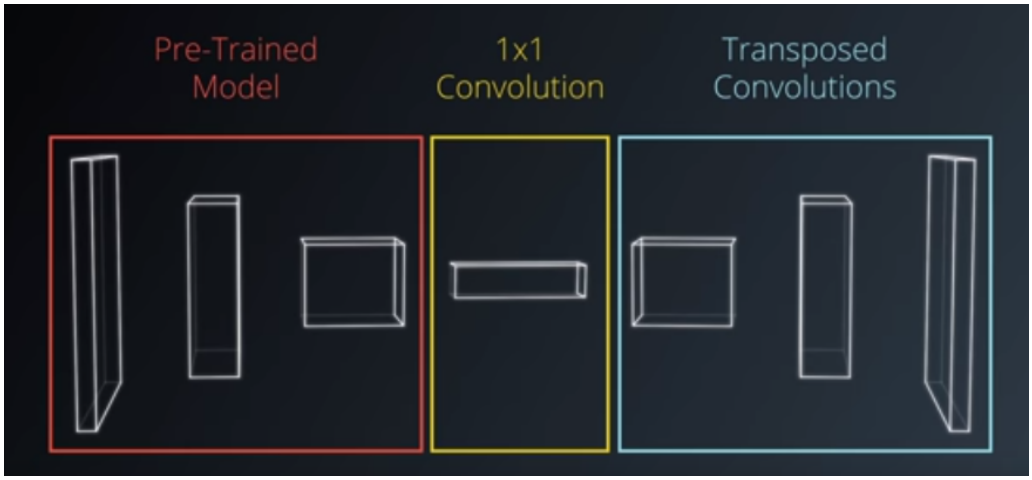
Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 160, 160, 3)	0
separable_conv2d_keras_1 (Se	(None, 80, 80, 64)	283
batch_normalization_1 (Batch	(None, 80, 80, 64)	256
separable_conv2d_keras_2 (Se	(None, 40, 40, 128)	8896
batch_normalization_2 (Batch	(None, 40, 40, 128)	512
conv2d_1 (Conv2D)	(None, 40, 40, 256)	33024
batch_normalization_3 (Batch	(None, 40, 40, 256)	1024
bilinear_up_sampling2d_1 (Bi	(None, 80, 80, 256)	0
concatenate_1 (Concatenate)	(None, 80, 80, 320)	0
separable_conv2d_keras_3 (Se	(None, 80, 80, 128)	43968
batch_normalization_4 (Batch	(None, 80, 80, 128)	512
separable_conv2d_keras_4 (Se	(None, 80, 80, 128)	17664
batch_normalization_5 (Batch	(None, 80, 80, 128)	512
bilinear_up_sampling2d_2 (Bi	(None, 160, 160, 128)	0
concatenate_2 (Concatenate)	(None, 160, 160, 131)	0
separable_conv2d_keras_5 (Se	(None, 160, 160, 64)	9627
batch_normalization_6 (Batch	(None, 160, 160, 64)	256
separable_conv2d_keras_6 (Se	(None, 160, 160, 64)	4736
batch_normalization_7 (Batch	(None, 160, 160, 64)	256
conv2d_2 (Conv2D)	(None, 160, 160, 3)	195

Total params: 121,721
Trainable params: 120,057
Non-trainable params: 1,664

Hiperparameter Tuning

- Optimizer : Adam, is a very famous optimization algorithm and usually outperforms the other possible choices.
- Batch Size : 32, is a typical number.
- Number of epochs : 30, this value is selected by trial and error but since the computational cost is high I decided to give a look to similar exercises and found out 30 would work ok.
- Learning Rate: 0.001, is a recommended learning rate for Adam.
- Steps per epoch : 100, the number of batches of samples before declarin an epoch is finished.
- Validation steps per epoch: 50, since data should be tested in every single epoch this number represents the number of images in validation over batch size.

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.



The encoder is a convolution network that reduces to a deeper 1x1 convolution layer (the feature extraction is hierarchical, you will find simple shapes and objects in the first layers and more complex objects and forms in the last ones), in contrast to a flat fully connected layer that would be used for basic classification of images. This difference has an advantage because it has the effect of preserving spacial information from the image, besides one of the advantages of 1x1 convolutions used within and the special type of separable convolutions we are using is that they are very computationally efficient.

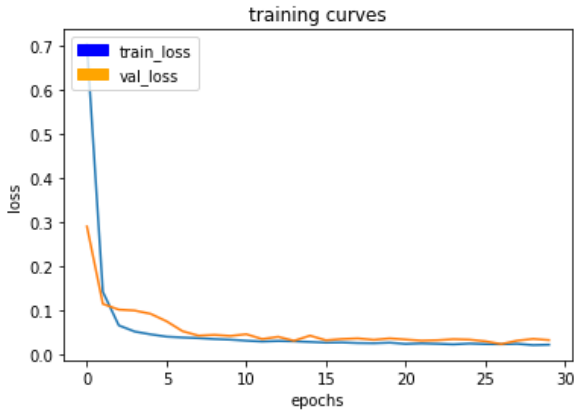
The decoder job is to perform upsampling layers to higher dimensions or resolutions. The decoder up scales the encoder output back into the dimensions of the original image.

The disadvantage of this encoder/decoder approach is that the decoder can’t reconstruct perfectly the image because the goal is to detect where in the image the object is located, not to reconstruct the same image.

Trainin Process

Unfortunatly I run out of AWS credits so I had to perform the training using my pc, after a very long period of time I could succesfully train the neural network with 30 epochs in aproximatly 20 hours.

```
100/100 [==============================] - 2139s - loss: 0.0210 - val_loss: 0.0351
Epoch 30/30
 99/100 [============================>.] - ETA: 18s - loss: 0.0219
```

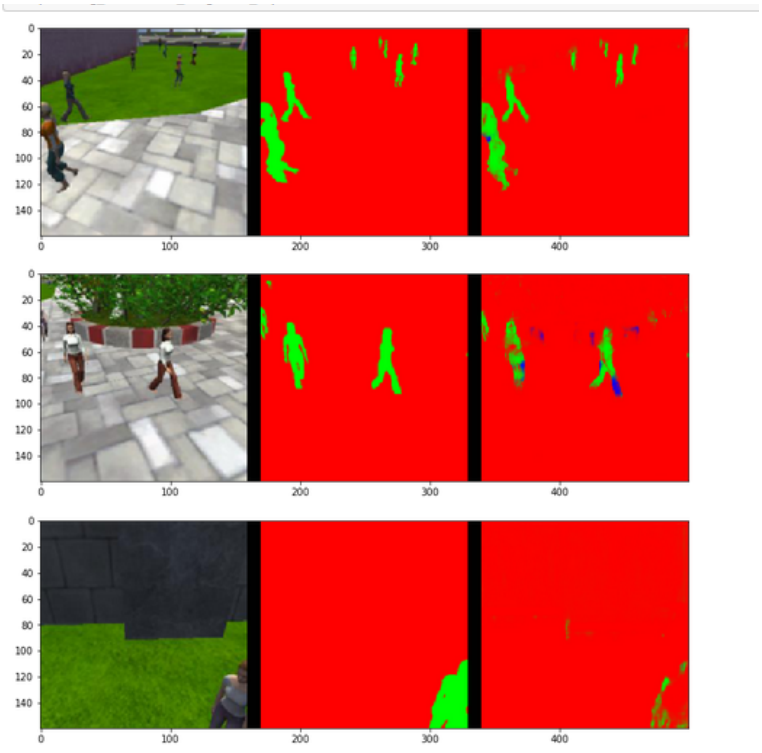



```
100/100 [==============================] - 2158s - loss: 0.0219 - val_loss: 0.0325
```

```
Out[9]: <tensorflow.contrib.keras.python.keras.callbacks.History at 0x7fb1277e2d30>
```

Experiments and Results

Since the computational resources are so high for fully convolutional neural networks I only did only one implementation, the metric showed a performance above 0.4 and that was very satisfying. The performance in simulation following the target was so also very good.



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.

In order to perform segmentation detecting other objects like cats or dogs this model is enough, I would have to retrain the algorithm using aditional data of those extra objects, depending of the number of classes detected the model would be more complex.

Future Enhancements

There ir room for improvement in this model's metric, is possible to get better performance using more data from people, try different architecture using more layers which let the network learn smaller features that can facilitate make predictions of the target object far away and parameter tuning like the number of epochs of weight initialization. Unfortunatly this network requires a lot of computational resources and would only be reasonable to do it using GPUs or AWS. I'm going to experiment even more as soon as I get extra credits on AWS.