*December 65, 2017*

*Udacity Robotics Deep Learning Project*

Cheshta Dhingra

1. Introduction

In this project, I built and trained a fully convolutional neural network to allow a drone to identify and follow a specific target – the ‘hero’ – within an image in a simulated environment. I was able to achieve a final IoU of ~42% (0.421).

1. Network Architecture

I chose to incorporate 2 encoder layers and 2 decoder layers, connected by a 1x1 convolutional layer in my fully convolutional network. The first encoding layer takes the image as raw input and finds simple structures within it, such as edges, using semantic segmentation (pixel by pixel). The second layer then takes the output of the first and gradually identifies more complex patterns, such as faces. This output becomes the input to the 1x1 convolutional layer, which reduces dimensionality similar to a fully connected layer, but it preserves spatial information. Finally, there are 2 decoder layers which essentially reverse the process of the encoder layers, and upscale the image back to its original dimensions. This is followed by a softmax activation function that finally returns the output.

* 1. Create an encoder\_block
  2. Create a decoder\_block
  3. Build the FCN consisting of 2 encoder blocks, a 1x1 convolution, and 2 decoder blocks. This step requires experimentation with different numbers of layers and filter sizes to build your model.

Input output

64

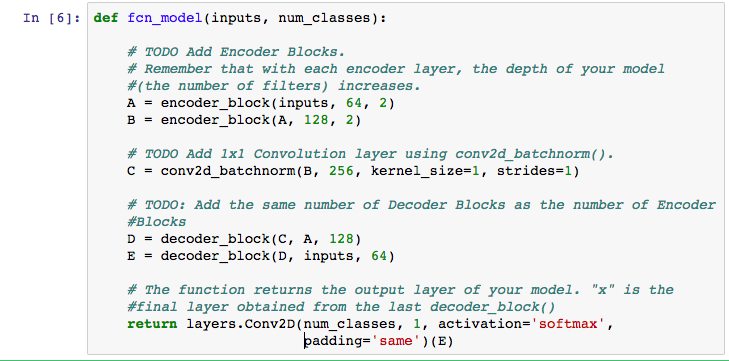
64

128

256

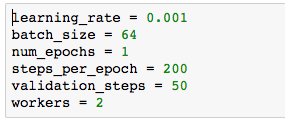
128

Encoder block 1x1 conv layer Decoder block



1. Parameter Tuning

I initially tried training on my local system but it took too long to even run one epoch, so I switched to the AWS GPU instance. Although I haven’t yet heard from Amazon about my own limit increase, luckily I was able to use a friend’s account, who had already requested a limit increase for another Udacity Nanodegree. I used a very small learning rate of 0.001 and this seemed to work well for all my tests. For the batch size, I initially tried 256 but this ended up taking too long so I reduced it to 64. I initially trained 2 epochs, which brought my loss down to about 0.09. I thought this was already quite good, but I decided to run it again with 10 epochs. I noticed the loss went down drastically to about 0.03 by the 2nd epoch, so I stopped it and trained it one last time with just 1 epoch. The steps per epoch, validation steps and workers were left at their default values of 200, 50 and 2 respectively. I finally got a loss of 0.0349 and a validation loss of 0.0431. I saved this model, which you will find in my submission.



1. Convolutional Layer

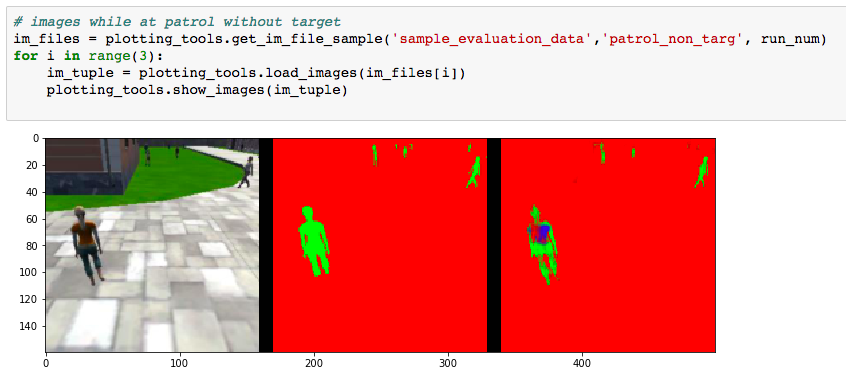
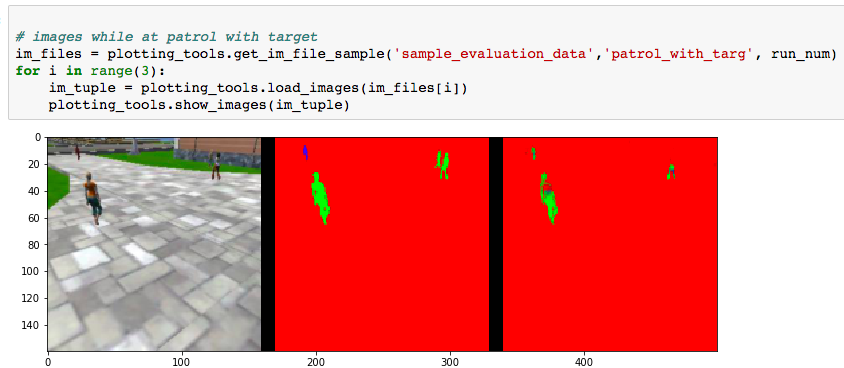
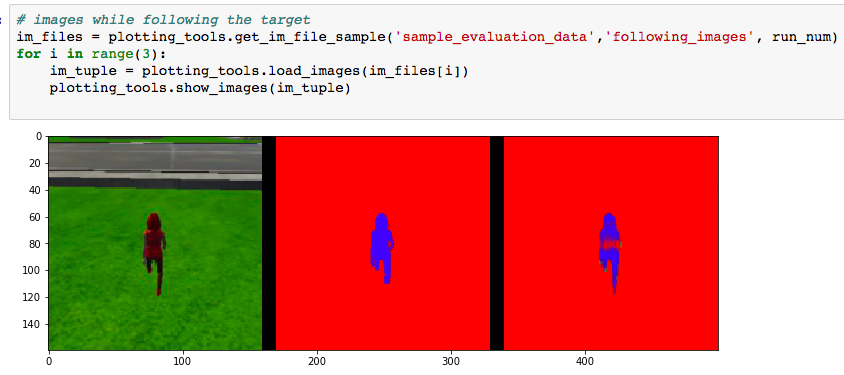
For this project, a fully connected neural network would not have worked, as spatial information would have been lost. Thus, we built a convolutional network, which preserves the spatial relationship between pixels by learning image features using small squares of input data. They are invariant to spatial translations, making our task of identifying targets within an image much more efficient.

1. Image Manipulation

In order to use this model and data to detect other objects (like a dog, or a car) instead of humans, we would need to train the network on images of the target objects.

1. Results

With my network I achieved a final IoU of 0.421 and a final grade score of 0.289. While the quad is following the target, I achieved a result of 0 false negatives and 0 false positives. However, I had several false positives while the quad is on patrol and the target is not visible as well as several false negatives when detecting the target from far away.



1. Limitations / Future enhancements

I am certain that if I collect my own data I will be able to achieve greater accuracy and efficiency; a much higher loss reduction and better overall score, as the volume of training data seems to be the predominant factor that determines the success of a neural network. In the same way that children learn to recognize objects by seeing them repeatedly, my network can become much more adept at identifying the hero if it is trained on more images of the hero in different possible scenarios. I could also continue to adjust the hyperparameters and/or add more layers to my network.

