RoboND "Follow Me" Project Report

Wolfgang Steiner
wolfgang.steiner@gmail.com

October 26, 2017

I. Introduction

In this project I trained a fully-convolutional neural network in order to perform semantic scene segmentation on images obtained from simulation. The aim is to distinguish a target object from the scene background and from other non-target objects in order to follow it with a robotic drone.

II. Network Architecture

The network architecture of the fully convolutional network used in this project is shown in Fig. 1. Following the input layer, three convolutional encoders reduce the size of the input image by a factor of two each, while the filter size of each layer increases in order to encode the feature vectors for the image segmentation. The encoders consist of a convolution followed by a batch normalization layer which ensures faster learning progress by re-normalizing the feature vectors for each batch during training. In the network with the highest accuracy the first encoder has a filter depth of 64 which is doubled for the deeper encoders.

The encoder layers are followed by a 1x1 convolution with a filter depth of 256. **This convolutional layer is used instead of a fully connected layer in order to retain the spacial information of the segmented image.** After this layer, three decoder layers are used to interpolate the segmentation back to the resolution of the input image. Each decoder uses bilinear interpolation to increase the resolution by a factor of two. Additionally, a second input is concatenated with the up-sampled tensor. This measure ensures that the network is able to do image segmentation at different scale or resolution levels.

The output layer consists of a dense layer with softmax activation function. It has three output classes for background, target object and other object. Thus the fully-connected output layer will be able to classify each of the pixels in the image as one of three output classes.

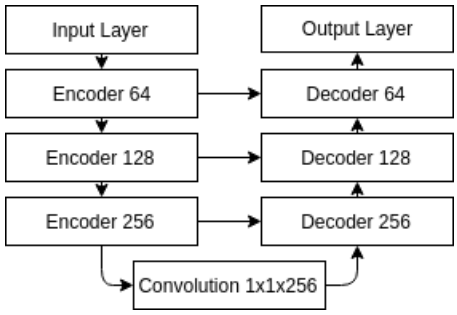

Figure 1: *Model architecture of the fully convolutional neural network.*

III. Training Procedure

a. Optimizer and Learning Rate

As an optimizer for training I chose the Adam algorithm which. The learning rate determines the step size during gradient descent. We would want the learning rate to be high in order to make fast progress during training. But if the learning rate is chosen too high, training might not converge. Choosing a learining rate that is too low might result in the training getting stuck in a local minimum which would not result in the best model possible. This problem is mitigated when using the Adam optimizer because it will dynamically adjust the training rate as the training progresses. From experience, I chose 0.001 as the training rate, which is also the default value for the Adam optimizer in Keras.

b. Training Epoch

The training epoch is an arbitrary convention used in the training of neural networks. Commonly one epoch is counted when all of the training examples

have been used once during optimization. The number of epochs is then a measure of how long the network has been trained on the training data. After each epoch, the validation loss is calculated by evaluating the validation images. I did not chose a specific number of epochs for training. I defined a maximum number of epochs of 64 for training. By using a checkpoint callback, the model was saved every time the validation loss decreased.

c. Batch Size

The batch size determines how many training examples are used in for one step of the optimization procedure. Here a balance must be achieved between convergence on the one hand which commonly favors smaller batch sizes and parallelization on the GPU which calls for larger batch sizes. I picked 32 as a batch size and found training performance to be good. Thus I did not have a reason to change or experiment with this value.

d. Data Augmentation

I implemented a simple data augmentation procedure which randomly flips the training and ground truth images horizontally. This kind of data augmentation is very easy to implement and practically doubles the number of training examples for free.

IV. Results

The results for different combinations of filter depth and training epochs are shown in Tab. 1. Here the filter depth of the first encoder layers are specified and the filter depth doubled for each succeeding layer. I started out with a filter size of 16/32/64 and trained the model for 16 epochs. The resulting accuracy score was 0.318. I then doubled the filter depth to 32/64/128 and achieved a score of 0.365.

At this point I introduced data augmentation by horizontal flipping and changed the training procedure by increasing the maximum number of training epochs while saving the model after each decrease of the validation loss. With these measures I was able to achieve a score of 0.445 after 52 epochs. Finally, I increased the filter depth to 64/128/256 and achieved a score of 0.466 after training the model for 61 epochs.

Table 1: *Model accuracy for different filter depth and number of training epochs.*

Filter Size	Num Epochs	Final IoU	Final Score
16	16	0.470	0.318
32	16	0.501	0.365
32	52	0.572	0.445
64	61	0.607	0.466

The learning curves of this final model is shown in Fig. 2. Some example images containing the target object and other objects are shown in Fig. 3.

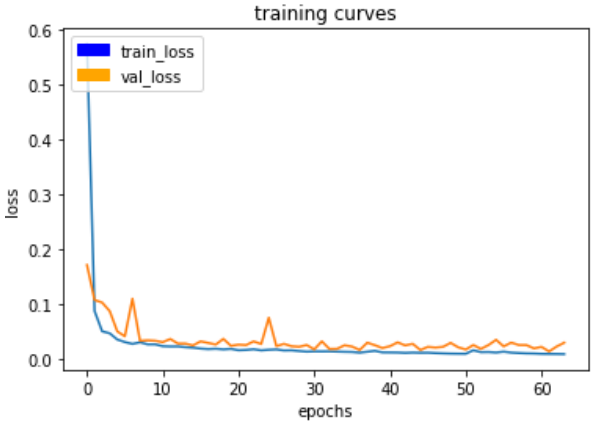

Figure 2: *Training curve of the final model with filter depth 64/128/256.*

V. Limitations of the Model

As this model was trained specifically on images derived from a simulator, I would not expect it to work particularly well on real-world images from a camera. Furthermore, the target object in the training data was a humanoid dressed in red clothes. Thus, it would not be possible for the drone to follow other persons dressed in different colors or even other objects or animals. We could use the same model architecture unchanged but train on different image data if the task was for the drone to follow one specific object. I think that it would also be possible to train the same model to classify different objects as a target but then the model would follow any of the object classified as "target" indiscriminately. If

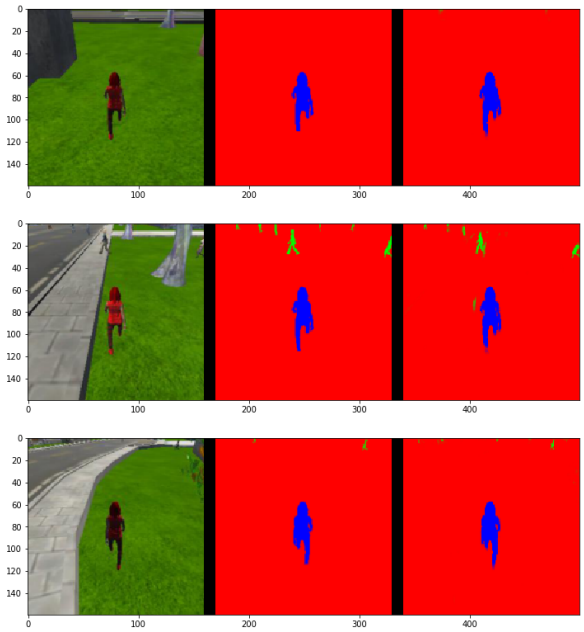

Figure 3: *Sample images from the test set: Original image (left), ground truth (center), segmentation results (right).*

the drone should be able to select one specific target object from a collection of targets, the number of classes would need to be altered. This could be achieved by changing the number of output neurons of the final dense layer of the model.

VI. Future Enhancements

In order to improve the accuracy of the model, the filter depth could be further increased for both the encoder and the decoder layers. Also it would be worthwhile to investigate whether additional layers would increase segmentation accuracy. Of course, collecting additional data would be advantageous as would be a more elaborate implementation of data augmentation. Images could be slightly rotated or skewed and the lighting could be adjusted on the training images.