Udacity Robotics Nanodegree

Project 4

Deep Learning Project: Follow Me

Frank Zerofsky

February 4, 2018

**Objective:**

The object of this project is the build a fully convolutional network (FCN) to apply a method called semantic segmentation to a data set. Semantic segmentation is used to both identify objects in a scene and their location within that scene. The ultimate goal is to have an aerial drone follow a specific target within an environment. This specific project identifies a hero target, being a person wearing a red shirt. The aerial drone must search for the target hero, and once found, follow the target.

**FCN Architecture:**

The fully convolutional network is made up of three main parts. The first part is an encoder where the input image is downsampled. The encoder blocks are fed into a 1x1 convolutional layer in order to preserve spatial data within the image. The output of the 1x1 convolutional layer is then upsampled in the decoder layers. The decoder layers upsample the image back to the original size.



A set of skip connections is also introduced to the network to add back in lost information within the image during downsampling/upsampling. The reasoning for this is to ….. The final layer of the network is a softmax relu activation where the final

**Data Collection:**

Additional data was collected within the simulator to augment the supplied data. Data was collected for the training set and the validation set. Each of the data sets were taken at different places within the simulator environment. Images of the path for the hero, aerial drone, and bystanders are shown below for both data collection sets.

**Model Training and Hyper Parameters:**

By using the Keras for tensorflow, the network parameters are able to be easily and quickly adjusted between training runs. The following hyper parameters below were adjusted to optimize the model..

Epochs: Epoch is one cycle forward and backward through the number of inputs set by the batch size. By increasing the number of epochs, the total amount of data seen by the network in increased.

Learning rate: When using stochastic gradient descent the learning rate is a constant value multiplied by the gradient of the loss value. This essentially effects the rate of change for the weights in the model. By having a smaller learning, the system will learn slower. The advantage to having a smaller learning rate allows the model to converge the loss function more efficiently.

Batch size: defined as the number of data inputs sent through the network in a single pass.

Steps per epoch: This hyper parameter describes the number of batches of training data to go through the network in a single epoch. This value is recommended to be set to the number of training data points divided by the batch size.

Validation steps per epoch: The same as steps per epoch, but it applies to the validation data set.

Workers: The number of workers allows more processes to work simultaneously. The default setting for this project was 2. This value was kept constant because the network training was performed on a single AWS EC2.xlarge instance.

The network’s hyper parameters were tuned using a brute force method. A starting point was picked in order to set a baseline, and one to two parameters were then changed between runs depending on the results of the previous run. Each training run hyper parameter and final scores are outlined below.

Training Run #1:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Learning Rate | Batch Size | Epochs | Steps | Val Steps | Workers | Final IoU Score |
| 0.1 | 32 | 10 | 200 | 50 | 2 | 0.335 |

The first training run was a test to make sure the network was working properly, and the AWS instance was setup properly. Low numbers for batch size and epochs were chosen, as to not spend time excessive time in a setup run. This run allowed a baseline to be set for further training refinement.

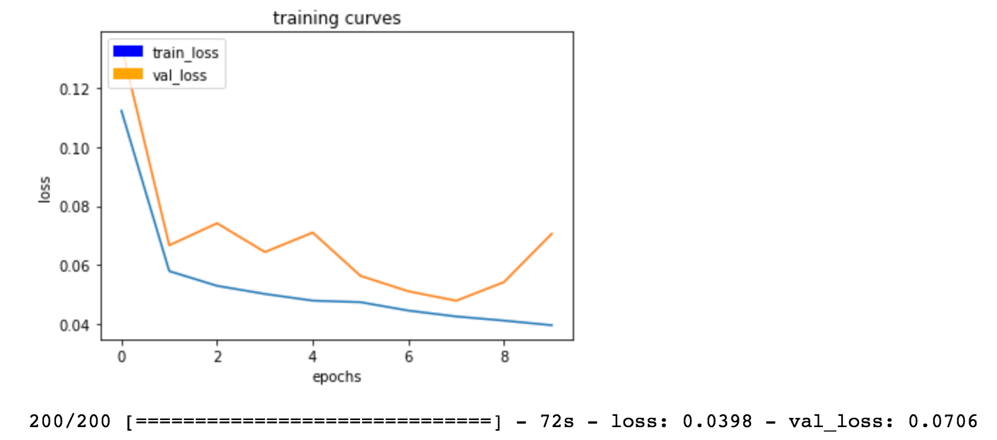
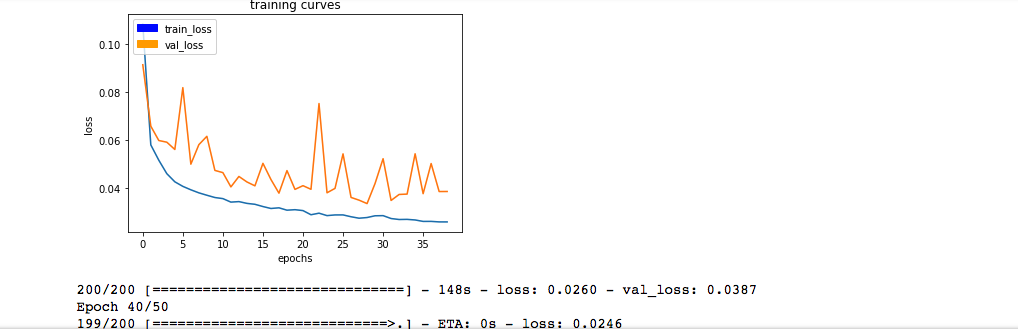


Figure 1: training results run 1 loss vs. epochs

Training Run #2:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Learning Rate | Batch Size | Epochs | Steps | Val Steps | Workers | Final IoU Score |
| 0.1 | 64 | 50 | 200 | 50 | 2 | 0.400 |

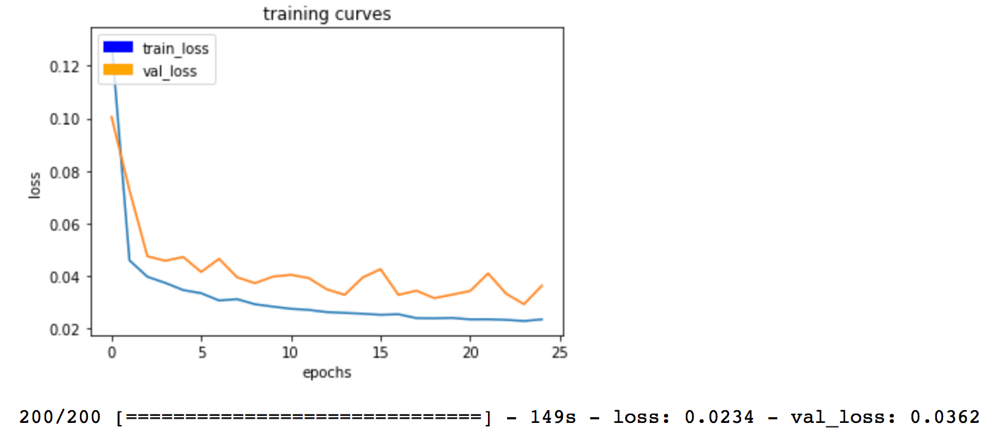
Based on the results of training run #1, the FCN did not have enough epochs to properly converge. Therefore, the epochs were increased from 10 to 50. By increasing the batch size, more images will be sent through the network in each epoch to allow further refined training results.



Based on the runtime results for run 2, the training was stopped at 40 epochs. This was based on the validation loss not converging.

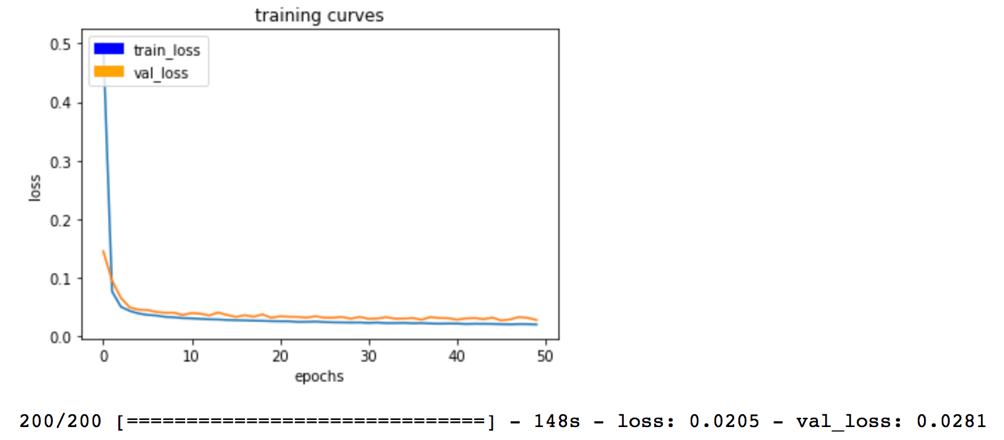
Training Run #3:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Learning Rate | Batch Size | Epochs | Steps | Val Steps | Workers | Final IoU Score |
| 0.01 | 64 | 25 | 200 | 50 | 2 | 0.407 |



Training Run #4:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Learning Rate | Batch Size | Epochs | Steps | Val Steps | Workers | Final IoU Score |
| 0.001 | 64 | 50 | 200 | 50 | 2 | 0.412 |



**Final Results:**

The final results of the network performance are determined using the intersection over union (IoU) method. The IoU is calculated as the average of the IoU values for when the aerial drone if following behind the target and the detection of the target hero from far away. This is

The final training run accomplished a 41.2% with the validation loss and the training loss values converged at approximately 0.02.

The model for the fully convolutional network could be used to identify a different object such as a dog, cat, or car. In order to successfully identify a different target other than the “hero”, the network would need to be retrained with a new dataset containing the desired target. Therefore, this specific project is only relatable to finding the hero in a red shirt, but with a new dataset could be used to find a new target.

**Files:**

GitHub repository: <https://github.com/fzero6/deeplearning-follow-project.git>

Python files:

model\_training.ipynb

follower.py

preprocess\_ims.py

Other files:

model\_weights.hd5

**References:**

1. Udacity, "Introduction to Robotics," [Online]. Available: <https://classroom.udacity.com/nanodegrees/nd209/parts/c199593e-1e9a-4830-8e29-2c86f70f489e>.
2. <https://github.com/udacity/RoboND-Rover-Project>