Project 4 - Follow Me

Network Architecture

In the project, we are asked to build the neural network can not only detect the kind of the objects but also tell where the object is, which means the outcome of the network should contain the spatial information. In this case, we need to build and train a Fully Convolutional Network (FCN) for following the moving target.

The Definitions of FCN

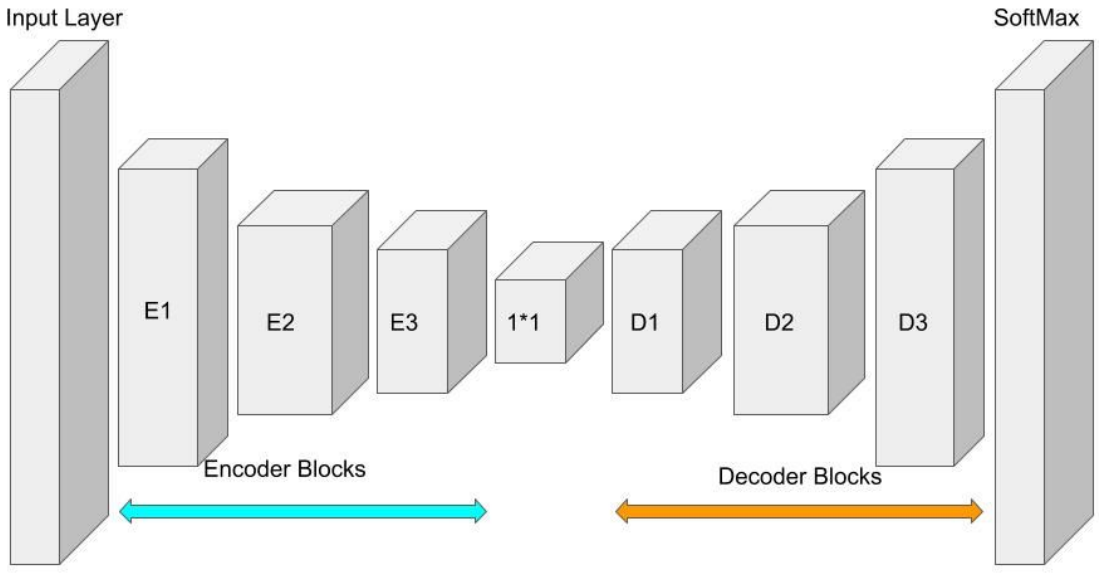


Fig 1. The definition of FCN

Compared to the normal CNN, FCN has following special definitions:

1. Encoder Blocks: The encoder in this project is a normal CNN utilize “separable convolution” and “batch normalization”. By using this kind of technique, we can improve the training rates and the network training speed.

2. Decoder Blocks: The task of the decoder is the almost inverse process of the encoder. In this project, an “upsampler” layer and a “concatenation” layer are included. The “upsampler” take the data from the adjacent pixels and return the average value of them. The “concatenate” layer combine the upsample and large input in series.
3. 1*1 convolutions: The 1*1 is used to reduce the dimensionality and retain the most of the data at the same time. In math, the 1*1 convolutions can be seen as an equivalent to Fully Connected Layers (FCLs). But 1*1 convolutions can process various inputs, which are more flexible, while FCLs need a fixed size.
4. Skip connections: The skip connections are used to improve the gradient flow through the network.

Model Building

The model we used in this project contains 3 parts:

1. Encoder blocks contain encoder layers.
2. 1*1 Convolution layer.
3. Decoder blocks contain decoder layers.

The part of this code is contained in “fcn_model(inputs, num_classes)”

Hyperparameters

To train the model well, we need to tune the hyperparameter carefully. Here we have 5 hyperparameters to tune:

1. Learning Rate: It is the step size that the model moving to the correct results. A higher learning rate usually means a faster training process. However, an improper learning rate may cause overfitting or underfitting. A low learning rate will result in a more accurate outcome but lead to longer training time. The starting learning_rate = 0.01, this result in a large pulse in the curve. Then a learning_rate = 0.005, the same thing happened again. Then a learning_rate = 0.004 is applied, the result becomes a lot of better. When a learning_rate = 0.003 is applied, the training time becomes incredibly long and not acceptable. So, here I set the ***learning_rate = 0.004***

2. Number of Epochs: The number of epochs is the number of forward and back propagations. In this project, I set the number of epochs as 40 by trial and error, because the model cannot be more accurate after 40 epochs. The epochs number of 20 and 30 are also tried but they didn't give a result as good as 40.
   So, here I set ***num_epochs = 40***
3. Steps per Epoch: This is the number of batches through each epoch.
   steps_per_epoch = 200
4. Validation Steps: This is the number of batches through each epoch in the validation set.
   validation_steps = 50
5. Batch Size: This is the number of images processed at the same time. Large batch size will make the training procedure faster but consume the more computational resource.
   batch_size = 32

There is also another hyperparameter, which is "workers". The "workers" is the number of processes to work on solving the problem. Here I set ***workers = 4***, since we use AWS.

The following figure shows the comparison between epoch(3), epoch(14), epoch(27) and epoch(40).

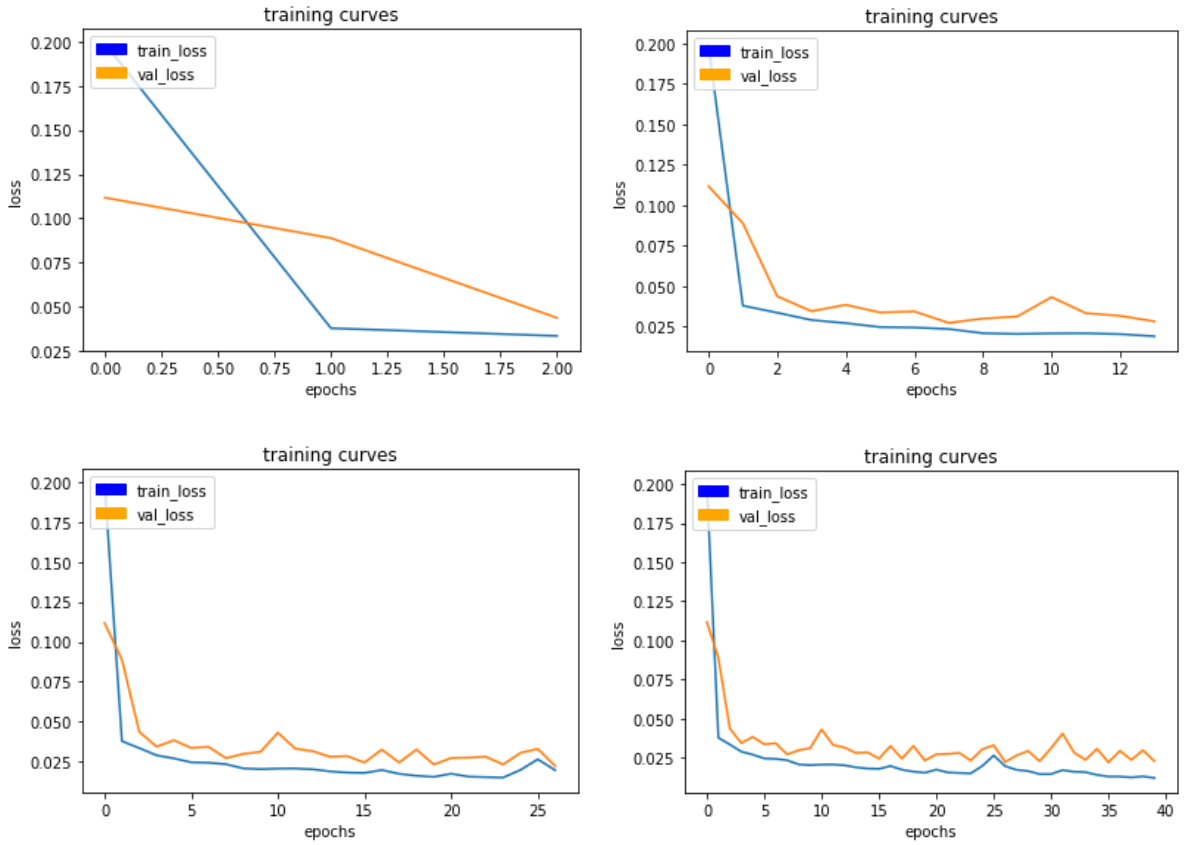
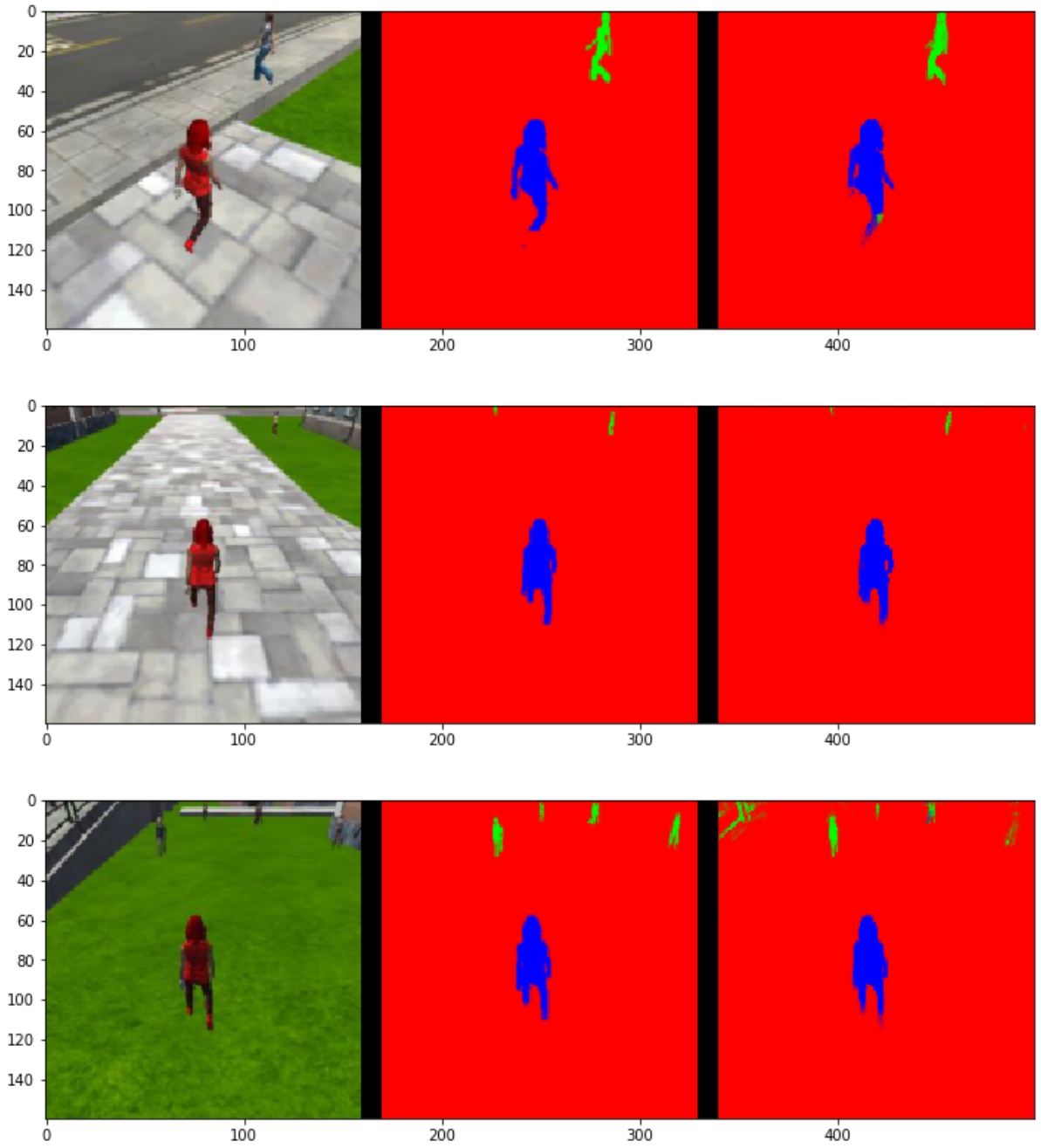


Fig.2 The result of the training curve from epoch(3), epoch(14), epoch(27) and epoch(40)

Results

The are 3 predictions:

Following the target:



number of validation samples intersection over the union evaulated on 542

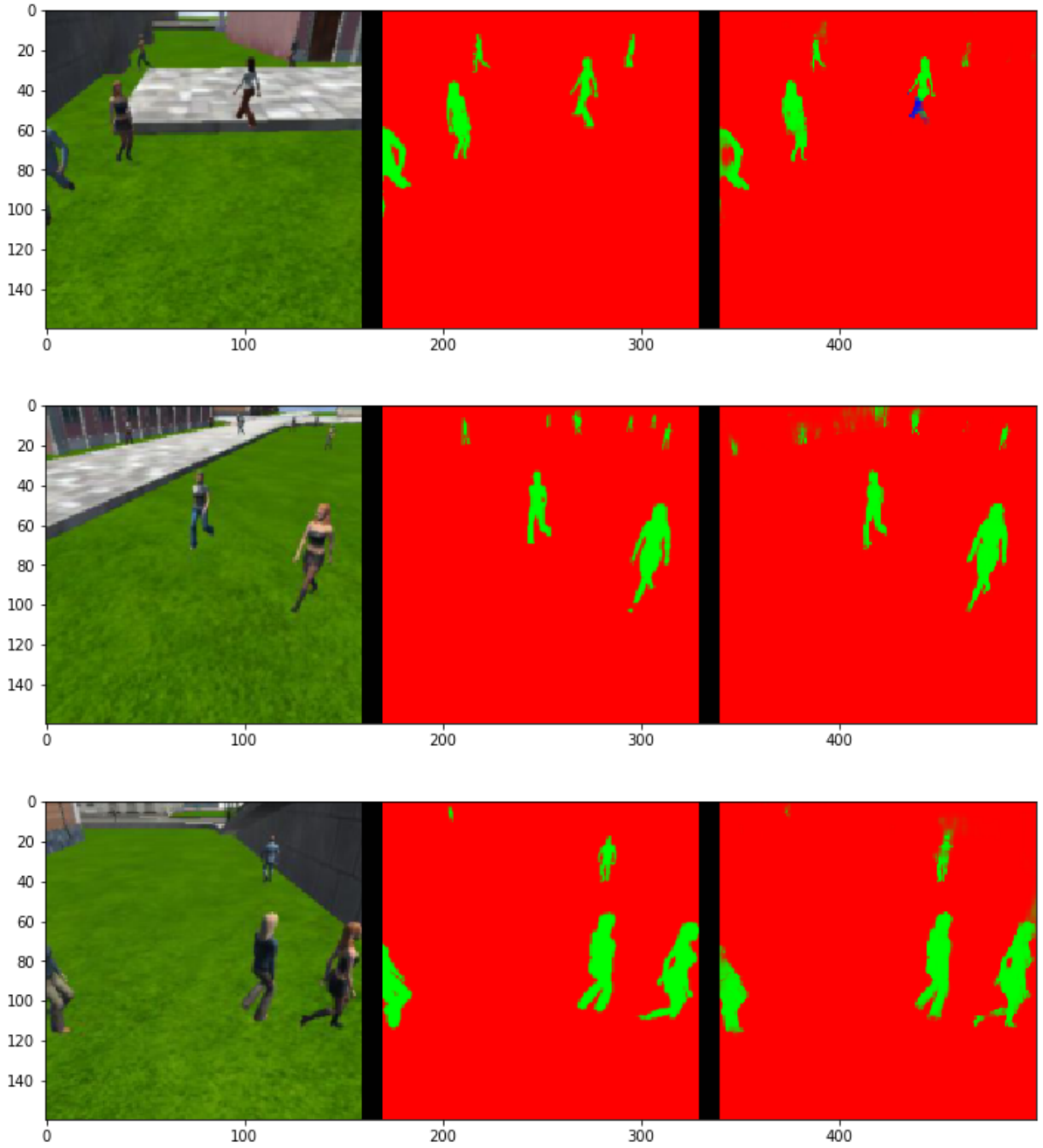
average intersection over union for background is 0.9959129001549901

average intersection over union for other people is 0.37639870793956864

average intersection over union for the hero is 0.9041799145507773

number true positives: 539, number false positives: 0, number false negatives: 0

Without target:



number of validation samples intersection over the union evaulated on 270

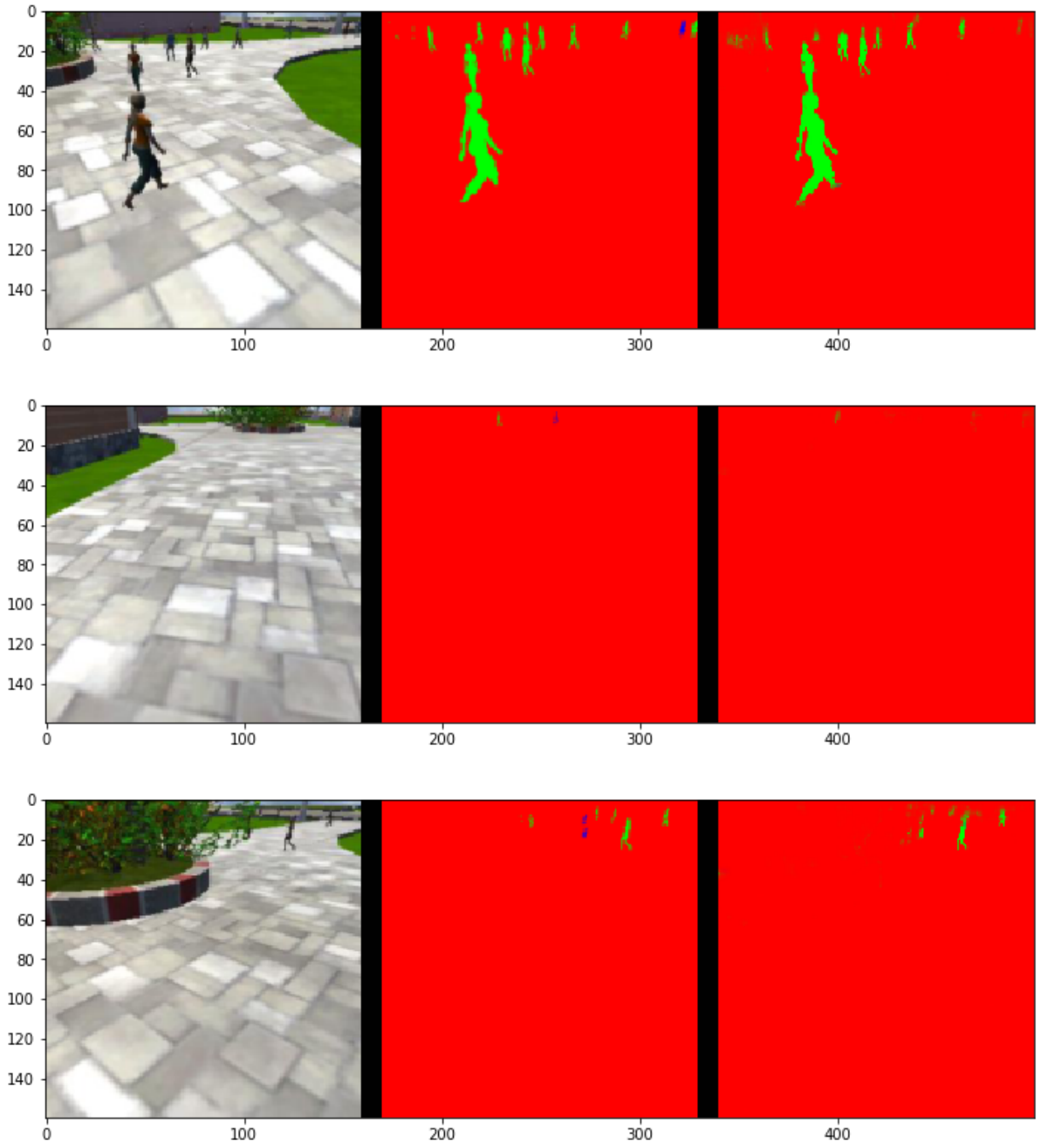
average intersection over union for background is 0.9885334509445189

average intersection over union for other people is 0.7748117991792716

average intersection over union for the hero is 0.0

number true positives: 0, number false positives: 39, number false negatives: 0

Target at a distance:



number of validation samples intersection over the union evaulated on 322

average intersection over union for background is 0.9969335281179194

average intersection over union for other people is 0.48057333752113635

average intersection over union for the hero is 0.25482256487773935

number true positives: 146, number false positives: 2, number false negatives: 155

Evaluation

The weight for the score is ***0.7775255391600454***

The IoU is ***0.579501239714***

The final score is the product of the "*weight"* and "*IoU"*

Final Score = *0.412473880649 > 0.40*

Future Improvements

First is the answer to the question of "whether this model and data would work well for following another object (dog, cat, car, etc)", is "No". The model needs to be re-trained using the data captured from the new desired object (dog, cat, car, etc). However, most of the code can be reused in training the new model. The hyperparameter we chose in this project can be a good guidance for training the model for other objects.

The improvement of this project is we can enlarge the dataset by adding more images of the objects. We can change the clothes of the target or the brightness of the environment to make the model universal.