UDACITY

PROJECT

Semantic Segmentation

A part of the Self-Driving Car Engineer Program

PROJECT REVIEW

CODE REVIEW 4

NOTES

SHARE YOUR ACCOMPLISHMENT!

Meets Specifications

Dear student, the network implementation is quite good. Skip Layer Architecture has been correctly implemented for Fully Convolutional Networks for Semantic Segmentation. Clearly shows how well the concepts have been grasped. Hope you had a fun learning experience doing the project.😄

Thank you for providing the detailed readme file. It was very helpful.

Some very important suggestions and resources have been provided throughout the review. Please do check them and experiment with the network accordingly. All the very best!

Also, here are some more interesting resources to further dive deeper into Semantic Segmentation.

- A 2017 Guide to Semantic Segmentation with Deep Learning
- Dense-Segmentation: Pyramid Scene Parsing (PSPnet)

Happy learning! ✌🏻

Build the Neural Network

✓

The function `load_vgg` is implemented correctly.

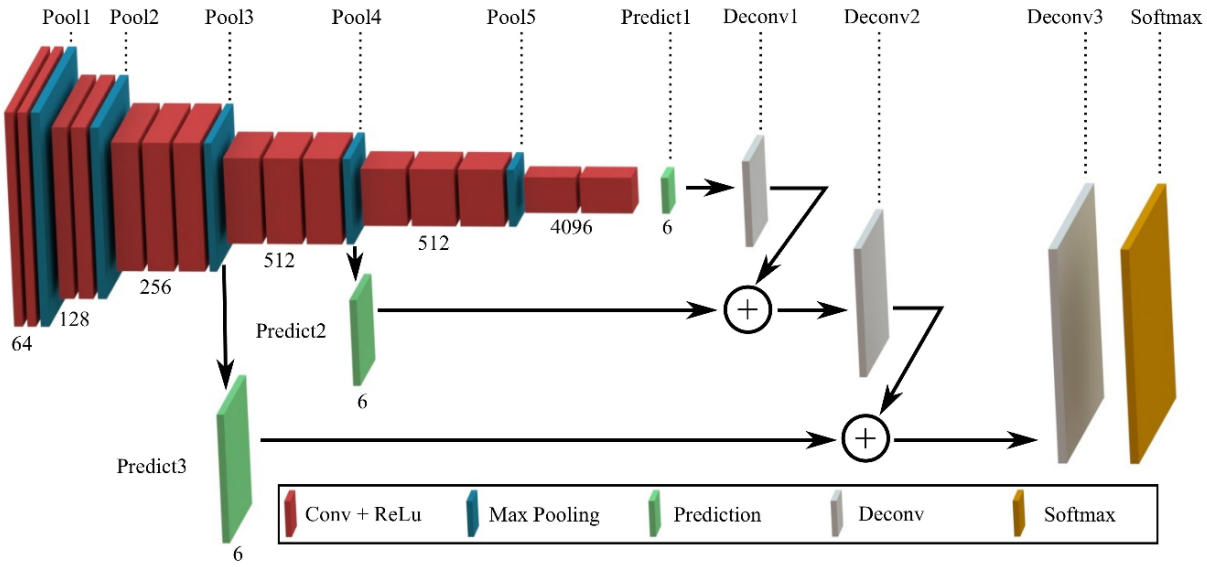
Good implementation of the function `load_vgg` to correctly

- load the pre-trained VGG model
- extract the required tensors from the loaded model
- return the extracted tensors as a tuple

Good job!

✓

The function `layers` is implemented correctly.



Original Skip Layer Architecture

Good implementation of the function `layers` to build the Skip Layer Architecture.

Positive points:

- Very good choice to use custom `kernel_initializer` to appropriately initialize the weights of the model.
- Also, very good choice to `regularize` the weights of the model using `l2_regularization`.
- Good job making all the skip connections properly. **Skip connections are found to improve the segmentation accuracy**, as discussed by the authors in the original paper.

Please do check these informative resources to study more about the significance of **weight initialization** and **regularization**.

✓

The function `optimize` is implemented correctly.

A brief and clean implementation of `optimize`.

Please check the **code review** for **important suggestions**.

✓

The function `train_nn` is implemented correctly. The loss of the network should be printed while the network is training.

Good choice to `print` the `loss` as the network trains.
`loss` provides useful insight into the network and helps work the errors out.

Neural Network Training

✓

The number of epoch and batch size are set to a reasonable number.

Fully Convolutional Networks are large and tend to occupy a lot of memory. For this reason, `batch_size` is advised to be kept to a small number.

- `batch_size = 8` and `epochs = 50` seem to be quite good keeping in mind the potential memory issues.
- Also, `learning_rate = 1e-4` seems to be decent enough and tunes pretty well.

Overall, good job tuning the hyperparameters 👍

Please do check this interesting resource: **Tradeoff batch size vs. number of iterations to train a neural network**

✓

On average, the model decreases loss over time.

✓

The project labels most pixels of roads close to the best solution. The model doesn't have to predict correctly all the images, just most of them.

A solution that is close to best would label at least 80% of the road and label no more than 20% of non-road pixels as road.

Awesome work!
The output produced by the network is quite good and the roads seem to be well segmented.

DOWNLOAD PROJECT

4 CODE REVIEW COMMENTS ›

RETURN TO PATH

Rate this review

Student FAQ