PROJECT

Object Classification

A part of the Deep Learning Nanodegree Foundation Program

PROJECT REVIEW | CODE REVIEW | NOTES

Meets Specifications

SHARE YOUR ACCOMPLISHMENT

Good job overall training a convolutional neural network to classify images from the CIFAR-10 dataset.

You put a lot of effort into this project!

Congratulations and good luck with your Nanodegree!

Required Files and Tests

✓ The project submission contains the project notebook, called “dlnd_image_classification.ipynb”.

✓ All the unit tests in project have passed.

Preprocessing

✓ The `normalize` function normalizes image data in the range of 0 to 1, inclusive.

✓ The `one_hot_encode` function encodes labels to one-hot encodings.

Neural Network Layers

✓ The neural net inputs functions have all returned the correct TF Placeholder.

✓ The `conv2d_maxpool` function applies convolution and max pooling to a layer.

The convolutional layer should use a nonlinear activation.

This function shouldn’t use any of the tensorflow functions in the tf.contrib or tf.layers namespace.

✓ The `flatten` function flattens a tensor without affecting the batch size.

✓ The `fully_conn` function creates a fully connected layer with a nonlinear activation.

To read more about the various activation functions please go to the following link:

http://cs231n.github.io/neural-networks-1/

Highly recommended!

✓ The `output` function creates an output layer with a linear activation.

Good job here!

Neural Network Architecture

✓ The `conv_net` function creates a convolutional model and returns the logits. Dropout should be applied to alt least one layer.

Good job implementing `conv_net` function!

For more detailed tips on practical convolutional network architectures please go to:

http://cs231n.github.io/convolutional-networks/#architectures

Rate this review

Neural Network Training

✓ The `train_neural_network` function optimizes the neural network.

✓ The `print_stats` function prints loss and validation accuracy.

✓ **The hyperparameters have been set to reasonable numbers.**

10 `epochs` looks good because the best number for this hyperparameter is such a number when the validation accuracy stop increasing.

Great! `batch_size` of 128 will do the job! The batch size can vary, depending on the performance of the computer. Higher is usually better. Anything less than 64 is incorrect.

Your `keep_probability` hyperparameter of 0.8 does the job!
If you set the `keep_probability` hyperparameter to 1.0 you aren't using dropout therefore it's incorrect.
The recommended value in this scenario is between 0.3 and 0.8

✓ **The neural network validation and test accuracy are similar. Their accuracies are greater than 50%.**

Great work getting the test and validation accuracies above 70%!

DOWNLOAD PROJECT

RETURN TO PATH

Student FAQ

Rate this review