UDACITY

PROJECT

Object Classification

A part of the Deep Learning Nanodegree Foundation Program

PROJECT REVIEW

CODE REVIEW

NOTES

SHARE YOUR ACCOMPLISHMENT!

Meets Specifications

Congratulations, your passed your object recognition project. With a couple of rubric items, I wrote some specific feedback. I encourage you to read it and may be experiment a little with your network. Good luck!

Required Files and Tests

The project submission contains the project notebook, called "dlnd_image_classification.ipynb".

All the unit tests in project have passed.

Well done, your code runs flawlessly.

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.

Excellent implementation using a global variable

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.

Well structured solution

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

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

Well done. Good idea to include the dropout into this function.

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

Neural Network Architecture

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

A good network that gets you an acceptable accuracy. Improvements could come from increasing the number of layers in both the conv and the fully connected part of your network. This article well describes what the role is of the various layers in a convolutional network.

Neural Network Training

The `train_neural_network` function optimizes the neural network.

The `print_stats` function prints loss and validation accuracy.

Indeed: keep_prob = 1.0 here

The hyperparameters have been set to reasonable numbers.

With these hyperparameters your network arrives at a useful accuracy. I suggest increasing the number of epochs to see if you can get the accuracy a little higher. Also the batch size is at the bare minimum. If your hardware allows: choose 256 or 512.

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

Well done!

DOWNLOAD PROJECT

RETURN TO PATH

Student FAQ