UDACITY

PROJECT

Object Classification

A part of the Deep Learning Nanodegree Foundation Program

PROJECT REVIEW

CODE REVIEW

NOTES

SHARE YOUR ACCOMPLISHMENT!

Meets Specifications

Great work on this project. You have met all the requirements of the project, and your network produces very good results.
Your project is accepted as complete, but please read the comments to see how you might be able to improve.
Good luck with the 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.

In your implementation, the resulting values will be in range [0.1, 0.9], not [0, 1] as required. You could have written it as: `return np.array(x)/255.`

This was the only issue in the project, so I will let it slide.

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

Good work on implementing `one_hot_encode`. You could have also used LabelBinarizer or OneHotEncoder from scikit-learn.

Neural Network Layers

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

You have correctly implemented all neural network input functions!

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.

Your implementation of `conv2d_maxpool` is great!

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

Good

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

Very good. Impressive that you decided not to use Layers and Layers (contrib) packages.

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

Perfect

Neural Network Architecture

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

Great! Your implementation of `conv_net` meets all the requirements.
You can read more on convolutional network architectures here: http://cs231n.github.io/convolutional-networks/#architectures and
http://stats.stackexchange.com/questions/148139/rules-for-selecting-convolutional-neural-network-parameters

Neural Network Training

The `train_neural_network` function optimizes the neural network.

You have correctly used session.run to perform NN training.

The `print_stats` function prints loss and validation accuracy.

Yes, you have correctly used `valid_features` and `valid_labels` and set keep_prob = 1!

The hyperparameters have been set to reasonable numbers.

Batch size of 128 is adequate.
Keep probability is in a good range. The number of epochs also seems to work fine.

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

Impressive results! Your test and validation accuracy are similar. It means that NN does not overfit to validation dataset and does not lose generalization on test dataset.

DOWNLOAD PROJECT

Rate this review

Student FAQ