UDACITY

PROJECT

Object Classification

A part of the Deep Learning Nanodegree Foundation Program

PROJECT REVIEW

CODE REVIEW

NOTES

SHARE YOUR ACCOMPLISHMENT!

Meets Specifications

Awesome

Congratulations on passing all the requirements for this project! You've done a good job at implementing the ConvNet and it produces fairly good results.

You could improve this project by building a more complex ConvNet. Here's a link to get you started:
http://rodrigob.github.io/are_we_there_yet/build/classification_datasets_results.html#43494641522d3130

You could also implement batch normalization. See the links:

1. https://www.tensorflow.org/api_docs/python/tf/nn/batch_normalization
2. https://www.quora.com/Why-does-batch-normalization-help

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.

Preprocessing

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

Good job here. Normalizing the inputs is a key step in the preprocessing stage.

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

Suggestion

You could also use sklearn's Label Binarizer to do this

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.

Good work here. The max pool, conv and ReLu have been implemented correctly. The weights are also initialized to small random numbers.

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

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

Good work in making this a nonlinear activation.

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.

Good job. The ConvNet also has dropout applied to each of the FC layers.

Neural Network Training

The `train_neural_network` function optimizes the neural network.

The `print_stats` function prints loss and validation accuracy.

The function prints the validation accuracy.

The hyperparameters have been set to reasonable numbers.

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

Excellent! The test and validation accuracies are well above the 50% mark.