PROJECT

Object Classification

A part of the Deep Learning Nanodegree Foundation Program

PROJECT REVIEW | CODE REVIEW | NOTES

Requires Changes

SHARE YOUR ACCOMPLISHMENT

1 SPECIFICATION REQUIRES CHANGES

Good project demonstrating understanding of building a convolutional classification network from components.

Required Files and Tests

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

The jupyter notebook does not contain execution results. The html file is being reviewed.

✓ **All the unit tests in project have passed.**

Preprocessing

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

As previous reviews.

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

As previous reviews.

Neural Network Layers

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

Well done. All placeholders are correctly instantiated.

Note: Placeholders are used to hold the input values to be used in a TensorFlow session. Placeholders can be viewed as in the same way as function parameters. Variables are used to hold values which can be updated in a TensorFlow session, in particular trainable values such as biases and weights.

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.

As previous reviews.

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

Well done in implementing `flatten(...)` using base tf functionality

Note: This functionality is also provided by the `tf.contrib` method `tf.contrib.layers.flatten(x_tensor, num_outputs)`.

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

It is better to specify the activation function for `tf.contrib.layers.fully_connected` as different library versions have differing results.

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

Well done specifying the activation function.

Neural Network Architecture

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

Nice architecture. Noted:

- small conv patch size
- small conv stride
- small pool size
- small pool stride

Networks following this pattern generally have high performance.

Reference articles: These article provides a detailed discussions convolutional network architecture:
https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/
http://cs231n.github.io/convolutional-networks/#architectures

Neural Network Training

✓ **The `train_neural_network` function optimizes the neural network.**

Well done. The session will evaluate the optimizer which in turn will minimize the cost (using softmax cross entropy) of the conv net.

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

Well done:

- setting the `keep_prob` to 1.0 for evaluating network performance
- training loss calculated on `feature_batch` and `label_batch`
- validation accuracy calculated on `valid_features` and `valid_labels`

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

epochs
There is room to reduce the number of epochs:

- at epoch 14 the validation accuracy reaches approx 54% - with training cost of approx 1.2
- at epoch XXX the validation accuracy is still approx 54% - with training cost of approx 0.9
- the training cost has declined for plateaued validation accuracy indicating overfitting to the training dataset

Suggestion: reduce training to point where validation accuracy plateaus

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

Well done, on completion of training the testing accuracy is 55%.

RESUBMIT PROJECT

DOWNLOAD PROJECT

Learn the best practices for revising and resubmitting your project.

RETURN TO PATH

Student FAQ