UDACITY

PROJECT

Object Classification

A part of the Deep Learning Nanodegree Foundation Program

PROJECT REVIEW

CODE REVIEW

NOTES

SHARE YOUR ACCOMPLISHMENT!

Meets Specifications

It's a remarkable job! Well done! Keep it up!

I suggest you check out the following Stanford CS231n resource on general advice about training neural networks. It has a number of great tips!

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! All unit tests 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.

Good job! Here's an alternative way (for the sake of learning):

```
def one_hot_encode(x):
  num_examples, num_classes = len(x), 10
  one_hot = np.zeros((num_examples, num_classes))
  one_hot[np.arange(num_classes), x] = 1
  return one_hot
```

Neural Network Layers

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

When defining `neural_net_image_input`, you could have written shape as `[None, *image_shape]`.

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.

Well done!

Note that you could have defined `fully_conn` in terms of `output` function since they only differ in `fully_conn` having a non-linear activation function. Hence, the following definition would suffice:

```
def fully_conn(x_tensor, num_outputs):
  return tf.nn.relu(output(x_tensor, num_outputs´))
```

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.

It's a descent architecture for this problem!

Neural Network Training

The `train_neural_network` function optimizes the neural network.

The `print_stats` function prints loss and validation accuracy.

Well done! Many students make a mistake here by computing training accuracy rather than 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%.

Good job! You managed to train a neural network, which beats an accuracy threshold of 50%.

DOWNLOAD PROJECT

RETURN TO PATH

Student FAQ