Requires Changes

**1 SPECIFICATION REQUIRES CHANGES**

This is a good first submission. You have a very good understanding of the underlying concepts. This isn't an easy assignment - we hope you've learned a lot about tensorflow and building a convolution neural network from scratch. Please look at my comments above on how to further improve your project and the performance of your model. All the best with your next submission.

**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.**

Well done! Your range of input should now be between 0 and 1. When inputs to the neural network are normalized, neural network training is often more efficient, which leads to a better predictor.

**The one\_hot\_encode function encodes labels to one-hot encodings.**

**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 implemented!

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

**The fully\_conn function creates a fully connected layer with a nonlinear activation.**

Required: Please apply a non-linear activation. You can again use the RELU activation as it works very well with images.

Suggestion: I would recommend the link below to understand more about different activation functions:  
<http://cs231n.github.io/neural-networks-1/>

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

Well done! Please note that since this is the output layer we shouldn't be applying any activation functions to this layer.

**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 connecting all the layers properly!

Suggestion:

* You can use multiple convolution and fully connected layers
* For the fully connected layer you can try more num of neurons (outputs) to increase its ability to learn. For example try 128, 256 or 512 neurons

**Neural Network Training**

**The train\_neural\_network function optimizes the neural network.**

**The print\_stats function prints loss and validation accuracy.**

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

Your choice of hyper parameter is good.  
The blog below has an interesting comparison of CIFAR 10 accuracies for different architectures:  
<http://zybler.blogspot.ca/2011/02/table-of-results-for-cifar-10-dataset.html>  
You can try reading through some of the papers and understanding what has worked well for others.

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