Neural Networks and Deep Learning

CSCI 5922 – Assignment 4 (Fall 2017) by **Akshit Arora (108631342)**

As per guideline on piazza, here is the code I started working with: https://github.com/Hvass-Labs/TensorFlow-Tutorials/blob/master/06_CIFAR-10.ipynb and it uses a module that is used for managing cifar10 dataset: https://github.com/Hvass-Labs/TensorFlow-Tutorials/blob/master/cifar10.py.

Part 1:

The intention for deciding how to split the data set into training and validation sets is to be able to compare two models.

In my case, I have chosen to take 20% of my data as validation set and the remaining is training set because it takes too long to do 1 learning experiment with 10,000 epochs. In order to make my training fast, I randomly select batches of 64 images from training data set and treat it as an iteration.

If I had better infrastructure (or some way to make my training faster):

- I could have done **5-fold cross validation** so that network is trained for every data point. By 5-fold cross validation I simply mean that I do 5 different learning experiments, each time a different 20% of the whole data set is selected as validation set and remaining is training set.
- And applied model averaging after the 5-fold cross validation.

Stopping Criterion: 10,000 epochs.

Part 2:

I conducted 3 classes of experiments:

0) To test out my current implementation of CNN. (no CV set defined)
1) Setting Model's Hyper-parameters (figuring out number of hidden units etc.) More training time needed to do 5-fold cross validation.
2) Studying effect of drop-out.

Experiment 0: (Code Available: 06_CIFAR-10.ipynb)

Objective: To test out the current implementation of CNN.

Input:

- Random batches of 64 from training images (32*32 pixels)
- Training images were pre-processed in the following ways:
    - Random cropping to (24*24)-pixel size

    - Random left/right flipping
    - Random Hue, Saturation, Contrast and Brightness
- **No cross-validation data set defined for this experiment**.

Network definition (2 conv2D each followed by maxpool layer; Then 2 fully connected layers; ReLU activation used):

1. Conv2D (Receptive field: 5*5*64; Batch normalization)
2. Max pool (5*5 size; Strides = 2)
3. Conv2D (Receptive field: 5*5*64)
4. Max pool (5*5 size; Strides = 2)
5. Flatten output
6. Fully Connected Layer (256 units)
7. Fully Connected Layer (128 units)
8. SoftMax classifier (10 units)

Results:

Training Accuracy: **85%**
Test Set: **74%**
CV Accuracy: **N/A**
Epochs: **10,000**

Experiment 1:

Objective: Setting Model's Hyperparameters. I intend to use 20% training data as validation set and use its accuracy to figure out adequate number of conv2d layers and FC layers. 10,000 epochs fixed.

Input:

- Same as previous section except 20% of training set is now cross validation set.

Summary of results:

#	Brief Architecture Definition	Max. Training set accuracy observed	Validation set accuracy	Test set accuracy
A	**2 conv2d** followed by max pooling and **2 FC layers**	71.9%	65.6%	65.2%
B	**4 conv2d*** followed by max pooling and **2 FC layers**	76.6%	64.1%	63.8%
C	**2 conv2d** followed by max pooling and **3 FC layers****	**84.4%**	**65.3%**	**64.7%**

**unable to use 4 or more FC layers since Jupyter Kernel would crash all the time!

*unable to use more than 4 conv2d layers since Jupyter Kernel crashes

Details of sub-experiments:

A) Let's start with 2 conv2D each followed by max-pool layer; Then 2 fully connected layers.
   a. Network definition (ReLU activation used):
      1. Conv2D (Receptive field: 5*5*64; Batch normalization)
      2. Max pool (5*5 size; Strides = 2)
      3. Conv2D (Receptive field: 5*5*64)
      4. Max pool (5*5 size; Strides = 2)
      5. Flatten output
      6. Fully Connected Layer (256 units)
      7. Fully Connected Layer (128 units)
      8. SoftMax classifier (10 units)
   b. Training set accuracy: **71.9%**
   c. Test set accuracy: **65.2%**
   d. Validation set accuracy: **65.6%**
   e. Code available: **06_CIFAR-10-q1.ipynb**

B) 4 conv2d layers instead of 2
   a. Network definition (same as before, 2 more conv2d added with max-pooling each)
   b. Training set accuracy:
   c. Test set accuracy:
   d. Validation set accuracy:
   e. Code available: **06_CIFAR-10-q2.ipynb**

C) 2 conv2d layers 3 fully connected layers
   a. Network definition:
      1. Conv2D (Receptive field: 5*5*64; Batch normalization)
      2. Max pool (5*5 size; Strides = 2)
      3. Conv2D (Receptive field: 5*5*64)
      4. Max pool (5*5 size; Strides = 2)
      5. Flatten output
      6. Fully Connected Layer (256 units)
      7. Fully Connected Layer (128 units)
      8. **Fully Connected Layer (128 units)**
      9. SoftMax classifier (10 units)
   b. Training set accuracy: **84.4%**
   c. Test set accuracy: **65.3%**
   d. Validation set accuracy: **64.7%**
   e. Code available: **06_CIFAR-10-q3.ipynb**

Experiment 2:

Objective: Study the effect of drop-out. 10,000 epochs fixed.

Input:

- Same as previous

Summary of Results:

#	Brief Architecture Definition	Max. Training set accuracy observed	Validation set accuracy	Test set accuracy
A	With Dropout	67.2%	60.8%	60.8%
B	Without Dropout	71.9%	65.6%	65.2%

Details:

A) With dropout:
   a. **Network Definition**
      1) Conv2D (Receptive field: 5*5*64; Batch normalization)
      2) Max pool (5*5 size; Strides = 2)
      3) Conv2D (Receptive field: 5*5*64)
      4) Max pool (5*5 size; Strides = 2)
      5) Flatten output
      6) Fully Connected Layer (256 units)
      7) Dropout (keep probability = 0.5)
      8) Fully Connected Layer (128 units)
      9) SoftMax classifier (10 units)
   b. Training set accuracy: **84.4%**
   c. Test set accuracy: **65.3%**
   d. Validation set accuracy: **64.7%**
   e. Code available: **06_CIFAR-10-q2-1.ipynb**

B) Without dropout:
   a. Network definition (ReLU activation used):
      1. Conv2D (Receptive field: 5*5*64; Batch normalization)
      2. Max pool (5*5 size; Strides = 2)
      3. Conv2D (Receptive field: 5*5*64)
      4. Max pool (5*5 size; Strides = 2)
      5. Flatten output
      6. Fully Connected Layer (256 units)
      7. Fully Connected Layer (128 units)
      8. SoftMax classifier (10 units)

b. Training set accuracy: **71.9%**
c. Test set accuracy: **65.2%**
d. Validation set accuracy: **65.6%**
e. Code available: **06_CIFAR-10-q1.ipynb**

Note: I learnt the implementation strategies (like using name scopes, helper functions for plotting layers etc.) with Tensorflow, from HVASS Tutorials (https://github.com/Hvass-Labs/TensorFlow-Tutorials) and therefore, my code looks very similar to the code used in their python notebook.

Also, I wanted to try more tricks with the architecture but I found myself limited to doing what the