Machine Learning
Homework 6

Karthik Nayak

Files: dataset.py

vnev: A virtual environment is provided in the folded named "HW6_venv" for easy use
use command: **. HW6_venv/bin/activate** to activate and then run python

Description: The program Convolutional Neural Networks to classify images of two different types of Corgi

Main libraries Used:

cv2	- Open CV is a library used to open and process images
numpy	- Fast matrix operation and storing image data, neural net data
tflearn	- provides a transparent higher-level API to TensorFlow
tensorflow	- open-source software used for machine learning applications

Workings:

Model Inputs

First the images from the training set are loaded using the given code.

Building Prediction Model

Then a multi-layer architecture consisting of alternating convolutions and nonlinearities are built using the tflearn library. These layers are followed by fully connected layers with rectified linear and softmax activations. The model is organized as follows:

convolution and rectified linear activation.

max pooling.

convolution and rectified linear activation.

max pooling.

fully connected layer with rectified linear activation.

Dropout Layer (k = 0.8)

fully connected layer with softmax activation.

Finally, an ADAM Optimizer used with cross-entropy as a common loss function.
The model graph can be seen on tensorboard

Model Training
Training is made very easy by tflearn as it is only a few lines of code.

'input' : training images
'targets' : training labels for each corresponding image

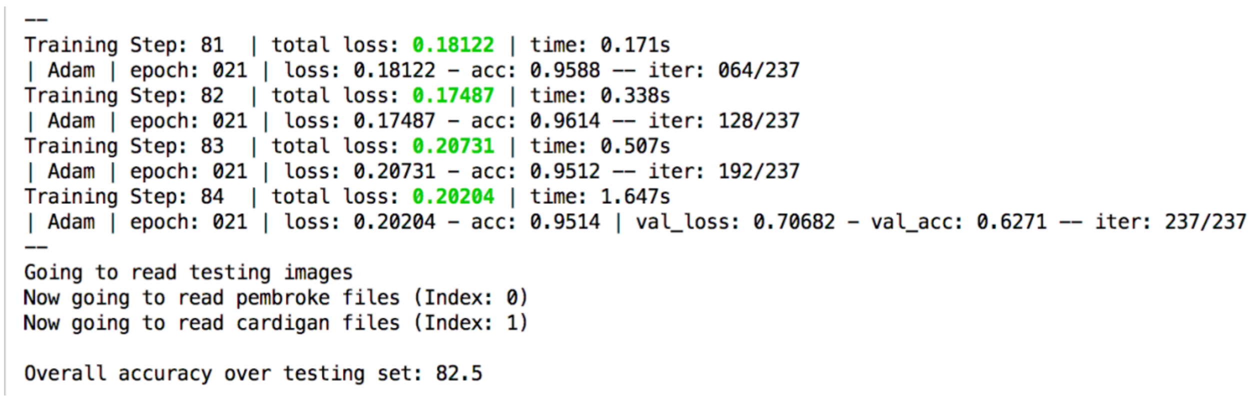
User decided parameters:
Learning Rate (alpha) = 0.001
n_epoch = 10

validation_set
'input' : validation images (subset of original training set that is removed)
'targets' : validation image labels for each corresponding image

Testing:
images from the testing set are loaded using openCV
Prediction is made for every image in testing set, by passing in the image in the previously trained model

Conclusion:
For the given architecture of the model and the parameters of alpha, epochs, etc. The CNN achieves accuracy of 70-85% accuracy over the images in the testing set.

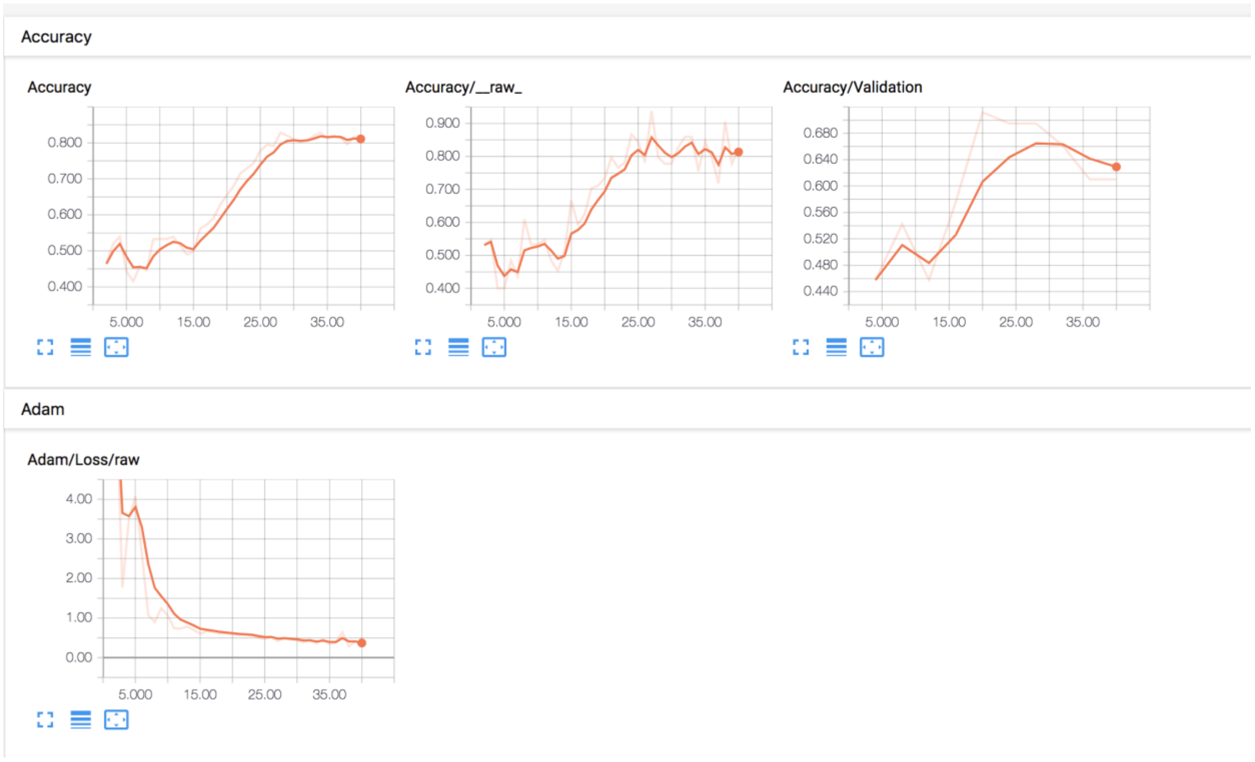
Final Output:



```
--
Training Step: 81  | total loss: 0.18122 | time: 0.171s
| Adam | epoch: 021 | loss: 0.18122 - acc: 0.9588 -- iter: 064/237
Training Step: 82  | total loss: 0.17487 | time: 0.338s
| Adam | epoch: 021 | loss: 0.17487 - acc: 0.9614 -- iter: 128/237
Training Step: 83  | total loss: 0.20731 | time: 0.507s
| Adam | epoch: 021 | loss: 0.20731 - acc: 0.9512 -- iter: 192/237
Training Step: 84  | total loss: 0.20204 | time: 1.647s
| Adam | epoch: 021 | loss: 0.20204 - acc: 0.9514 | val_loss: 0.70682 - val_acc: 0.6271 -- iter: 237/237
--
Going to read testing images
Now going to read pembroke files (Index: 0)
Now going to read cardigan files (Index: 1)

Overall accuracy over testing set: 82.5
```

Tesnsorboard Preview:



Citations

Harrison. “TFLearn - High Level Abstraction Layer for TensorFlow Tutorial.” *Python Programming Tutorials*, pythonprogramming.net/tflearn-machine-learning-tutorial/.

Damien, Aymeric. “TFLearn: Deep Learning Library Featuring a Higher-Level API for TensorFlow.” *TFLearn | TensorFlow Deep Learning Library*, tflearn.org/.