Poornima Andukuri

Introduction:

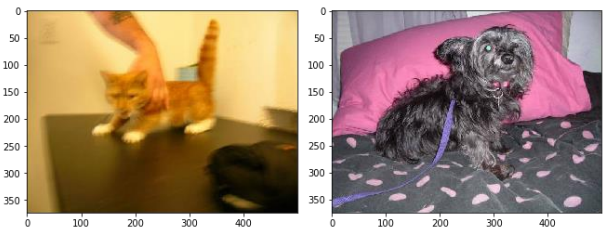
Assignment 7 uses the application of convolutional networks to assess the classification performance accuracy and processing times to predict the images provided. Use Tensorflow library to build a model that along with accurately predicting the image, will run efficiently on a larger data set.

The steps involved in the following analysis are

1. Use Tensorflow to build a 'Deep Neural Network' (DNN) model
2. Use train data set to fit and evaluate the various designs of the model using hidden layers
3. Use test data set to predict the images
4. Record the processing time and model accuracy on train and test datasets

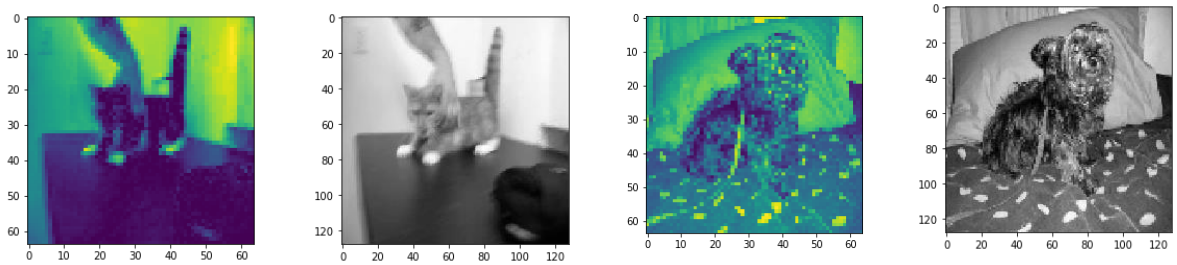
Exploratory Data Analysis:

There were two data sets provided. The first data set is train, that is used to create the model and the second data set is to test the model and predict the image. The train data has 25000 images split into 12500 pictures of cats and 12500 pictures of dogs. The pictures below are of the first in the list of cat and dog pictures in color and greyscale.



Data Preparation and Overview of Programming:

The first step in data preparation is to read the files, parse and resize the images in color and grayscale. The below are the pictures of the first cat and dog in the list of images



afterresized. For color image, the picture is resized to 64X64 and for grayscale it is resized to 128X128.

The data above is created by creating two different arrays for pictures of cats and dogs. We then concatenate the resized image into one two-dimensional array with an array of values of [1,0] for cats and [0.1] for dogs. The scikit train_test_learn method is used to split the final data to create train and validation models. We use TensorFlow library to fit four different neural networks with different number of convolutional filters, different filter sizes using conv_2d. Max pooling 2d is used with multiple values for kernel_size, along with different values of number of units for layers using fully_connected and 0.8 probability that each element is kept using dropout.

Model 1

```
Training Step: 3129  | total loss: 0.19961 | time: 101.186s
| Adam | epoch: 010 | loss: 0.19961 - Accuracy: 0.9159 -- iter: 19968/20000
Training Step: 3130  | total loss: 0.19160 | time: 108.272s
| Adam | epoch: 010 | loss: 0.19160 - Accuracy: 0.9211 | val_loss: 0.60347 - val
_acc: 0.8072 -- iter: 20000/20000
--
Wall time: 17min 41s
datetime.timedelta(seconds=1060, microseconds=386133)
```

Model 2

```
Training Step: 6259  | total loss: 0.17684 | time: 99.245s
| Adam | epoch: 020 | loss: 0.17684 - acc: 0.9624 -- iter: 19968/20000
Training Step: 6260  | total loss: 0.16500 | time: 106.586s
| Adam | epoch: 020 | loss: 0.16500 - acc: 0.9646 | val_loss: 0.76112 - val_acc:
0.7950 -- iter: 20000/20000
--
Wall time: 34min 54s
datetime.timedelta(seconds=2094, microseconds=118625)
```

Model 3

```
Training Step: 3129  | total loss: 0.61393 | time: 257.460s
| SGD | epoch: 010 | loss: 0.61393 - acc: 0.6690 -- iter: 19968/20000
Training Step: 3130  | total loss: 0.61451 | time: 272.474s
| SGD | epoch: 010 | loss: 0.61451 - acc: 0.6693 | val_loss: 0.60127 - val_acc:
0.6746 -- iter: 20000/20000
--
Wall time: 44min 36s
datetime.timedelta(seconds=2675, microseconds=423713)
```

Model 4

```
Training Step: 3129  | total loss: 0.76386 | time: 286.611s
| Adam | epoch: 010 | loss: 0.76386 - acc: 0.4787 -- iter: 19968/20000
Training Step: 3130  | total loss: 0.76045 | time: 304.256s
| Adam | epoch: 010 | loss: 0.76045 - acc: 0.4855 | val_loss: 0.75170 - val_acc:
0.5030 -- iter: 20000/20000
--
Wall time: 50min 37s
datetime.timedelta(seconds=3035, microseconds=726307)
```

Insights & Conclusions:

The recommendation is to use model 2 which uses 5 layers, 32, 64 and 128 convolutional filters of size 5, with kernel size of 5 and 1024 number of units for fully_connected. The accuracy of test predictions is at 96% and takes about 35 minutes to create, fit and validate the model using the train and test data.

Appendix:

The ipynb notebook and an html version of the notebook along with the output and Kaggle submission scores are included in the submission.