HW 6

Start Assignment

Due Mar 3 by 11:59pm **Points** 10 **Submitting** a text entry box or a file upload
Available Feb 23 at 12am - Mar 3 at 11:59pm 9 days

Part 1:

1. What are the advantages of a CNN over a fully connected deep neural network for image classification?

2. Why would you want to add a max pooling layer rather than a convolutional layer with the same stride?

3. When would you want to add a local response normalization layer?

4. Test below CNN codes with MNIST data set and show the model accuracy.

5. Make comments on your results in step 4.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
get_ipython().magic(u'matplotlib inline')


# In[2]:


import tensorflow as tf
from keras.layers import Input, Dense
from keras.models import Model


# In[3]:







# In[4]:


from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, Flatten
from keras.optimizers import Adam
from keras.layers.normalization import BatchNormalization
from keras.utils import np_utils
from keras.layers import Conv2D, MaxPooling2D, ZeroPadding2D, GlobalAveragePooling2D
from keras.layers.advanced_activations import LeakyReLU
from keras.preprocessing.image import ImageDataGenerator

np.random.seed(25)


# In[5]:


(X_train, y_train), (X_test, y_test) = mnist.load_data()
print("X_train original shape", X_train.shape)
print("y_train original shape", y_train.shape)
print("X_test original shape", X_test.shape)
print("y_test original shape", y_test.shape)


# In[6]:
```

```
plt.imshow(X_train[0], cmap='gray')
plt.title('Class '+ str(y_train[0]))


# In[7]:


X_train = X_train.reshape(X_train.shape[0], 28, 28, 1)
X_test = X_test.reshape(X_test.shape[0], 28, 28, 1)

X_train = X_train.astype('float32')
X_test = X_test.astype('float32')

X_train/=255
X_test/=255

X_train.shape


# In[8]:


number_of_classes = 10

Y_train = np_utils.to_categorical(y_train, number_of_classes)
Y_test = np_utils.to_categorical(y_test, number_of_classes)

y_train[0], Y_train[0]


# In[9]:


# Three steps to Convolution
# 1. Convolution
# 2. Activation
# 3. Pooling
# Repeat Steps 1,2,3 for adding more hidden layers

# 4. After that make a fully connected network
# This fully connected network gives ability to the CNN
# to classify the samples

model = Sequential()

model.add(Conv2D(32, (3, 3), input_shape=(28,28,1)))
model.add(Activation('relu'))
BatchNormalization(axis=-1)
model.add(Conv2D(32, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))

BatchNormalization(axis=-1)
model.add(Conv2D(64,(3, 3)))
model.add(Activation('relu'))
BatchNormalization(axis=-1)
model.add(Conv2D(64, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Flatten())
# Fully connected layer

BatchNormalization()
model.add(Dense(512))
model.add(Activation('relu'))
BatchNormalization()
model.add(Dropout(0.2))
model.add(Dense(10))

# model.add(Convolution2D(10,3,3, border_mode='same'))
# model.add(GlobalAveragePooling2D())
model.add(Activation('softmax'))


# In[10]:
```

```
model.summary()


# In[11]:


model.compile(loss='categorical_crossentropy', optimizer=Adam(), metrics=['accuracy'])


# In[12]:


gen = ImageDataGenerator(rotation_range=8, width_shift_range=0.08, shear_range=0.3,
 height_shift_range=0.08, zoom_range=0.08)

test_gen = ImageDataGenerator()


# In[13]:


train_generator = gen.flow(X_train, Y_train, batch_size=64)
test_generator = test_gen.flow(X_test, Y_test, batch_size=64)
```

```
# In[ ]:


# model.fit(X_train, Y_train, batch_size=128, nb_epoch=1, validation_data=(X_test, Y_test))

model.fit_generator(train_generator, steps_per_epoch=60000//64, epochs=5,
 validation_data=test_generator, validation_steps=10000//64)


# In[ ]:


score = model.evaluate(X_test, Y_test)
print()
print('Test accuracy: ', score[1])


# In[ ]:


predictions = model.predict_classes(X_test)

predictions = list(predictions)
actuals = list(y_test)

sub = pd.DataFrame({'Actual': actuals, 'Predictions': predictions})
sub.to_csv('./output_cnn.csv', index=False)
```

Part 2:

Test CNN over the cifar10 data set, which contains 32x32 colour images from 10 classes:

1. Use the below code to load the data set.

```
from keras.datasets import cifar10
(x_train, y_train), (x_test, y_test) = cifar10.load_data()

print("Train samples:", x_train.shape, y_train.shape)
print("Test samples:", x_test.shape, y_test.shape)
```

2. Show the 10 classes

```
NUM_CLASSES = 10
cifar10_classes = ["airplane", "automobile", "bird", "cat", "deer",  "dog", "frog", "horse", "ship", "truck"]

# show random images from train
cols = 8
rows = 2
fig = plt.figure(figsize=(2 * cols - 1, 2.5 * rows - 1))
for i in range(cols):
for j in range(rows):
random_index = np.random.randint(0, len(y_train))
```

```
ax = fig.add_subplot(rows, cols, i * rows + j + 1)
ax.grid('off')
ax.axis('off')
ax.imshow(x_train[random_index, :])
ax.set_title(cifar10_classes[y_train[random_index, 0]])
plt.show()
```

3. Define a CNN architecture and train your own model by playing with the network setup: like, performs convolution, performs 2D max pooling, changing activation function from ReLU to LeakyReLU, adding dropout etc.

```
# import necessary building blocks
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Activation, Dropout
from keras.layers.advanced_activations import LeakyReLU
```

```
```