EN2550 2018: Assignment 04

November 25, 2018

Please note that you must implement the key Python functions and scripts on your own. If you copy from the Internet or others, you will not get the learning experience intended through this assignment.

1. Study the BoVW model, e.g., by reading this post `https://kushalvyas.github.io/BOV.html`. Have this code running `https://github.com/kushalvyas/Bag-of-Visual-Words-Python`. I have uploaded a version that I slightly changed to work in Windows and Pyton 3. Answer the following questions:
   (a) List the important steps in he BoVW.
   (b) What is he size of the vocabulary?
   (c) What is the accuracy?
   (d) What is the feature detector used?
   (e) How would you use SURF? State code snipped that you changed.
   (f) What is the accuracy with SURF?
2. I have uploaded a piece of code for MNIST hand-written character recognition that uses Keras.
   (a) Sketch the network in the given file.
   (b) Create a CNN-based network with at least two convolutional layers, max-pooling, and two dense layers. Use max-pooling and drop-out.
   (c) Report the number of parameters, accuracy, and loss for the above.
   (d) Adapt one of the networks to work with CIFAR10.
   (e) Report the number of parameters, accuracy, and loss for the above.

Upload a four-page report named as your_index_a04.pdf. The report must include important parts of code, image results, and comparison of results . The interpretation of results and the discussion are important in the report. Extra-page penalty is 2 marks per page.

Listing 1: mnist.py

```
import matplotlib.pyplot as plt
import numpy as np
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras import backend as K

from helpers import *

batch_size = 16
num_classes = 10
epochs = 2

# input image dimensions
img_rows, img_cols = 28, 28
```

```
# the data, split between train and test sets
(x_train, y_train), (x_test, y_test) = mnist.load_data()


class_names = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']

# Showing a few examples
show_image_examples(class_names, x_train, y_train)

if K.image_data_format() == 'channels_first':
    x_train = x_train.reshape(x_train.shape[0], 1, img_rows, img_cols)
    x_test = x_test.reshape(x_test.shape[0], 1, img_rows, img_cols)
    input_shape = (1, img_rows, img_cols)
else:
    x_train = x_train.reshape(x_train.shape[0], img_rows, img_cols, 1)
    x_test = x_test.reshape(x_test.shape[0], img_rows, img_cols, 1)
    input_shape = (img_rows, img_cols, 1)

# Pick every 100th sample to speed-up (Set this to 1 in the final run.)
step = 100
x_train = x_train[::step, :, :]
y_train = y_train[::step]
x_train = x_test[::step, :, :]
y_train = y_test[::step]



x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
print('x_train␣shape:', x_train.shape)
print(x_train.shape[0], 'train␣samples')
print(x_test.shape[0], 'test␣samples')


# convert class vectors to binary class matrices
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)

model = Sequential()
model.add(Flatten(input_shape=input_shape))
model.add(Dense(512, activation='sigmoid'))
model.add(Dense(num_classes, activation='softmax'))


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

model_info = model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test))

model.summary()

score = model.evaluate(x_test, y_test, verbose=0)
```

```
print('Test␣loss:', score[0])
print('Test␣accuracy:', score[1])
plot_model_history(model_info)
```

Listing 2: helpers.py

```
# Adapted from http://parneetk.github.io/blog/cnn-cifar10/
import matplotlib.pyplot as plt
import numpy as np

def plot_model_history(model_history):
    fig, axs = plt.subplots(1,2,figsize=(15,5))
    # summarize history for accuracy
    axs[0].plot(range(1,len(model_history.history['acc'])+1),
                model_history.history['acc'])
    axs[0].plot(range(1,len(model_history.history['val_acc'])+1),
                model_history.history['val_acc'])
    axs[0].set_title('Model␣Accuracy')
    axs[0].set_ylabel('Accuracy')
    axs[0].set_xlabel('Epoch')
    axs[0].set_xticks(np.arange(1,len(model_history.history['acc'])+1),
                      len(model_history.history['acc'])/10)
    axs[0].legend(['train', 'val'], loc='best')
    # summarize history for loss
    axs[1].plot(range(1,len(model_history.history['loss'])+1),
                model_history.history['loss'])
    axs[1].plot(range(1,len(model_history.history['val_loss'])+1),
                model_history.history['val_loss'])
    axs[1].set_title('Model␣Loss')
    axs[1].set_ylabel('Loss')
    axs[1].set_xlabel('Epoch')
    axs[1].set_xticks(np.arange(1,len(model_history.history['loss'])+1),
                      len(model_history.history['loss'])/10)
    axs[1].legend(['train', 'val'], loc='best')
    plt.show()

def show_image_examples(class_names, features, labels):
    print(len(class_names))
    num_classes = len(class_names)
    fig = plt.figure(figsize=(8, 3))
    for i in range(num_classes):
        ax = fig.add_subplot(2, 5, 1 + i, xticks=[], yticks=[])
        idx = np.where(labels[:] == i)[0]
        features_idx = features[idx, ::]
        img_num = np.random.randint(features_idx.shape[0])
        im = features_idx[img_num, ::]
        ax.set_title(class_names[i])
        plt.imshow(im)
    plt.show()
```