Artificial Intelligence
Exercise 7

Marc Toussaint
Machine Learning & Robotics lab, U Stuttgart
Universitätsstraße 38, 70569 Stuttgart, Germany

3. Februar 2020

These is an "extra" exercise for students that need to collect extra points.

1 Keras for image classification

Keras is a really easy to use layer on top of Tensor Flow for neural networks in Python. Overview (details are found below):

(a) Use the pre-trained VGG16 neural network (that is included in Keras) to train and test on the CIFAR10 data set (that is included in Keras). To this end, use only the fully convolutional parts of the VGG16 (which can be applied to images of any size), fix the weights of this VGG16, add a fully connected output layer that maps to 10 classes, and train only this output layer. There are many projects online that do exactly the same. We give pointers and a starting point below.

(b) Test the change in performance when you change the output layer's activation function, or add two output layers.

(c) Remove again the output layer and use the VGG16 to map all images in the database to a feature vector. This creates a new data base with tuples (feature-vector, label).

(d) Use python's scikit-learn to train a logistic regression on this dataset. Compare the results with the original VGG16 results.

Abgabe: Please check in your python code in your group's git repository. Send me an Email. In this email, tell me your group number and report on two numbers: the classification error you get for (a) and for (c). Deadline: Please Feb 13 latest.

Details:

- Read https://keras.io/, perform the installation using

  `pip3 install keras --user`

  and test the "30 seconds" example
- Follow https://keras.io/datasets/ to test loading the cifar10 dataset
- On https://github.com/mjiansun/cifar10-vgg16 there is an example for training the VGG16 architecture on cifar10. Here are is one version that extracted the essential parts from that:

```
# load the pre-trained VGG16 without top layers
from keras.applications.vgg16 import VGG16
model = VGG16(include_top=False, weights='imagenet', input_tensor=None, input_shape=(32,32,3), poolin

# freeze all these layers
for l in model.layers:
    l.trainable = False
```

```
# add a fully connected top layer
from keras.layers import Flatten, Dense
from keras.engine import Model
x = Flatten()(model.layers[-1].output)    #a trivial 'reshape' from image to vector
pred = Dense(10, activation='sigmoid')(x) #a dense layer with 10 outputs

# report on the architecture
full_model = Model(model.input, pred)
full_model.summary()
```

- Prepare the data a bit:

```
from keras.datasets import cifar10
from keras.utils import np_utils

(X_train, y_train), (X_test, y_test) = cifar10.load_data()

X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train /= 255
X_test /= 255

Y_train = np_utils.to_categorical(y_train, 10)
Y_test = np_utils.to_categorical(y_test, 10)


N_TRAIN_SAMPLES = 1000
N_TEST_SAMPLES = 100
X_train = X_train[:N_TRAIN_SAMPLES, :, :, :]
Y_train = Y_train[:N_TRAIN_SAMPLES, :]
y_train = y_train[:N_TRAIN_SAMPLES]

X_test = X_test[:N_TEST_SAMPLES, :, :, :]
Y_test = Y_test[:N_TEST_SAMPLES, :]
y_test = y_test[:N_TEST_SAMPLES, :]

# check class balance
import numpy as np
print(np.sum(Y_train, axis=0))
print(np.sum(Y_test, axis=0))
```

- And finally train:

```
from keras import optimizers

full_model.compile(loss=categorical_crossentropy',
              optimizer=optimizers.SGD(lr=1e-3, momentum=0.9),
              metrics=['accuracy'])

full_model.fit(X_train, Y_train,
              batch_size=64,
              epochs=10,
              validation_split=0.1,
              shuffle=True)
```

```
scores = full_model.evaluate(X_test, Y_test, verbose=1)
print('Test loss:', scores[0])
print('Test accuracy:', scores[1])
```

- Continue with (b)-(c) on your own. Information on extracting features is here: https://keras.io/applications/, and information on logistic regression of scikit learn is here: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html.