Laboratory #3 Real time analysis and Pytorch

Table of Contents

Assignment Project Exam Help

In this lab we will work on three different applications of DNN. First we start with how to get access to webcam and how to send information to your DNN. On second part, we will see a RNN example and on last part of the lab we get familiar with a new for developing deep learning algorithms.

https://powcoder.com

The majority of second part is taken from: https://machinelearningmastery.com/.

Add WeChat powcoder

Step1. OpenCV and object detection

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code [http://www.opencv.org]. In this section, we want to install opencv and use it to design an object detection tool using tensorflow as backend.

Start with installing opencv package on python. If you are using Anaconda, simply use environment to install the package.

Or you can use following command inside your notebook:

```
!pip install opencv-python
```

Or you can use Anaconda prompt and type:

```
conda install -c conda-forge opencv
```

For more information about installation, you can follow these links:

For Mac:

https://www.learnopencv.com/install-opencv3-on-macos/

For Windows:

https://www.learnopencv.com/install-opencv3-on-windows/

1.1. Video capturing

Let's start with a very simple code. This code reads your webcam and shows the result. You can use "q" to exit the page.

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

```
import cv2
import numpy as np

cap = cv2.VideoCapture(0)

while True:
    ret, frame= cap.read() # Forever it returns the frame and ret which is
false or true
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY) #if you want to convert
the color
    cv2.imshow('frame', frame)
    cv2.imshow('gray', gray)    # to show the gray video

    if cv2.waitKey(1) & 0xFF == ord('q'):    # If q is pressed stop
        break

cap.release()
cv2.destroyAllWindows()
```

Q1- Try to understand each line of the code and explain it in your report.

1.2. Digit recognition

Q2- Use the code provided (mnist-cnn) to design a digit recognition but this time read the image of the number directly from your webcam. After the model was learnt, you can use following command to save the model for future uses:

```
model.save('address/to/file/my_model.h5')
```

This model is already trained and the result is saved as part of the lab package. To load the provided model you can use following command:

```
import tensorflow
new_model = tensorflow.keras.models.load_model('address/to/file/my_model.h5')
```

Now use the explained method in section 1.1 to read an image from your webcam and show the predicted output on screen.

You may use following function to get a counter of image:

```
def get_img_contour_thresh(img):
    x, y, w, h = 0, 0, 300, 300
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    blur = cv2.GaussianBlur(gray, (35, 35), 0)
    ret, thresh1 = cv2.threshold(blur, 70, 255, cv2.THRESH_BINARY_INV +
cv2.THRESH_OTSU)
    thresh1 = thresh1[y:y + h, x:x + w]
    contours, hierarchy = cv2.findContours(thresh1, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)[-2:]
    return img, contours, thresh1
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Figure 1- One suggested output

Now, let's call this function and make a real-time digit recognition:

```
import tensorflow
```

```
new_model = tensorflow.keras.models.load_model('C:/My
Courses/Spring2020/ANLY535/Lab3/my_model.h5')
# Handwritten recognition using camera
import cv2
import numpy as np

cap = cv2.VideoCapture(0)
ret, img = cap.read()
while (cap.isOpened()):
    ret, img = cap.read()
    ret
    img, contours, thresh = get_img_contour_thresh(img)
    ans1 = ''
    if len(contours) > 0:
        contour = max(contours, key=cv2.contourArea)
        if cv2.contourArea(contour) > 2500:
            # print(predict(w_from_model,b_from_model,contour))
            x, y, w, h = cv2.boundingRect(contour)
            # newImage = thresh[y - 15:y + h + 15, x - 15:x + w +15]
            newImage = thresh[y:y + h, x:x + w]
            newImage = cv2.resize(newImage, (28, 28))
            newImage = np.array(newImage)
            newImage = newImage.flatten()
            newImage = newImage.reshape(newImage.shape[0], 1)
            newImage2 = newImage.flatten().reshape(1,28,28,1)
            newImage2 = newImage2.astype('float32')
            newImage2 /= 255
            result = new_model.predict(newImage2)
            ans1= np.argmax(result)
            #ans1 = Digit_Recognizer_LR.predict(w_LR, b_LR, newImage)

    x, y, w, h = 0, 0, 300, 300
    cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 2)
    cv2.putText(img, "Prediction : " + str(ans1), (10, 320),
cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)

    cv2.imshow("Frame", img)
    cv2.imshow("Contours", thresh)
    k = cv2.waitKey(10)
    if k == 27:
        break

cap.release()
cv2.destroyAllWindows()
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

1.3. Face recognition

Now we want to design a face recognition code. Similar to what we did in character recognition, the very first step is to train a DNN based on some faces and eyes. To make our job easier, we can use two xml files that are trained version of faces and eyes. These two files are uploaded on Moodle. The source of the code and images samples can be found here: http://www-personal.umich.edu/~shameem/

You may use following code to write a sample face and eye recognition.

```
import cv2

face_cascade =
cv2.CascadeClassifier('address/to/file/haarcascade_frontalface_default.xml')
eye_cascade = cv2.CascadeClassifier('address/to/file/haarcascade_eye.xml')

cap = cv2.VideoCapture(0)

while True:
    ret, img = cap.read()
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, 1.3, 5)

    for (x,y,w,h) in faces:
            cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
            roi_gray = gray[y:y+h, x:x+w]
            roi_color = img[y:y+h, x:x+w]
            eyes = eye_cascade.detectMultiScale(roi_gray)
            for (ex,ey,ew,eh) in eyes:
                cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2)
    cv2.imshow('Lab 3 Face recognition',img)
    k = cv2.waitKey(30) & 0xff
    if k == 27:
        break

cap.release()
cv2.destroyAllWindows()
```

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Step2. RNN and text classification

IMDB sentimental analysis movie review is one of the famous datasets for analyzing the sentiments in reviews. This dataset contains the information of 25,000 review of movies. If you are interested in seeing the actual dataset, you may visit this website: http://ai.stanford.edu/~amaas/data/sentiment.

```
import numpy as np
from tensorflow.keras.datasets import imdb
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import LSTM
from tensorflow.keras.layers import Embedding
from tensorflow.keras.preprocessing import sequence
# fix random seed for reproducibility
np.random.seed(7)
```

For this part of lab, we are using the IMDB dataset preloaded in Keras. Based on Keras document section: "Dataset of 25,000 movies reviews from IMDB, labeled by sentiment (positive/negative). Reviews have been preprocessed, and each review is encoded as a sequence of word indexes

(integers). For convenience, words are indexed by overall frequency in the dataset, so that for instance the integer "3" encodes the 3rd most frequent word in the data."

The *imdb.load_data()* function allows you to load the dataset in a format that is ready for use in neural network and deep learning models. The words have been replaced by integers that indicate the ordered frequency of each word in the dataset. The sentences in each review are therefore comprised of a sequence of integers. We are constraining the dataset to the top 5,000 words. You can download the dataset using:

```
top_words = 5000
(X_train, y_train), (X_test, y_test) = imdb.load_data(num_words=top_words)
```

We need to truncate and pad the input sequences so that they are all the same length for modeling.

```
# truncate and pad input sequences
max_review_length = 500
X_train = sequence.pad_sequences(X_train, maxlen=max_review_length)
X_test = sequence.pad_sequences(X_test, maxlen=max_review_length)
```

Assignment Project Exam Help

Q3- We need to do word embedding at this point. What is the meaning of "*word embedding*" in context of NLP?

https://powcoder.com

The first layer is the Embedded layer that uses 32 length vectors to represent each word. The next layer is the LSTM layer with 100 memory units (smart neurons). Finally, because this is a classification problem we use a Dense output layer with a single neuron and a sigmoid activation function to make 0 or 1 predictions for the two classes (good and bad) in the problem.

Add WeChat powcoder

```
# design model
embedding_vecor_length = 32
model = Sequential()
model.add(Embedding(top_words, embedding_vecor_length,
input_length=max_review_length))
model.add(LSTM(100))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])
print(model.summary())
model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=3,
batch_size=64)
```

```
Epoch 10/20
391/391 [==============================] - 40s 101ms/step - loss: 0.0964 - accuracy: 0.9654 - val_loss: 0.5078 - val_accuracy: 0.8615
Epoch 11/20
391/391 [==============================] - 40s 102ms/step - loss: 0.0752 - accuracy: 0.9749 - val_loss: 0.5037 - val_accuracy: 0.8486
Epoch 12/20
391/391 [==============================] - 40s 101ms/step - loss: 0.1062 - accuracy: 0.9648 - val_loss: 0.5817 - val_accuracy: 0.8590
Epoch 13/20
391/391 [==============================] - 40s 102ms/step - loss: 0.1100 - accuracy: 0.9613 - val_loss: 0.5004 - val_accuracy: 0.8580
Epoch 14/20
391/391 [==============================] - 39s 101ms/step - loss: 0.0700 - accuracy: 0.9776 - val_loss: 0.5567 - val_accuracy: 0.8512
Epoch 15/20
391/391 [==============================] - 39s 101ms/step - loss: 0.0519 - accuracy: 0.9833 - val_loss: 0.6252 - val_accuracy: 0.8570
Epoch 16/20
391/391 [==============================] - 39s 101ms/step - loss: 0.0488 - accuracy: 0.9850 - val_loss: 0.6341 - val_accuracy: 0.8513
Epoch 17/20
391/391 [==============================] - 40s 101ms/step - loss: 0.0372 - accuracy: 0.9893 - val_loss: 0.7293 - val_accuracy: 0.8593
Epoch 18/20
391/391 [==============================] - 40s 102ms/step - loss: 0.0346 - accuracy: 0.9899 - val_loss: 0.6847 - val_accuracy: 0.8365
Epoch 19/20
391/391 [==============================] - 40s 101ms/step - loss: 0.0488 - accuracy: 0.9850 - val_loss: 0.6828 - val_accuracy: 0.8318
Epoch 20/20
391/391 [==============================] - 40s 102ms/step - loss: 0.0321 - accuracy: 0.9900 - val_loss: 0.7024 - val_accuracy: 0.8466
```

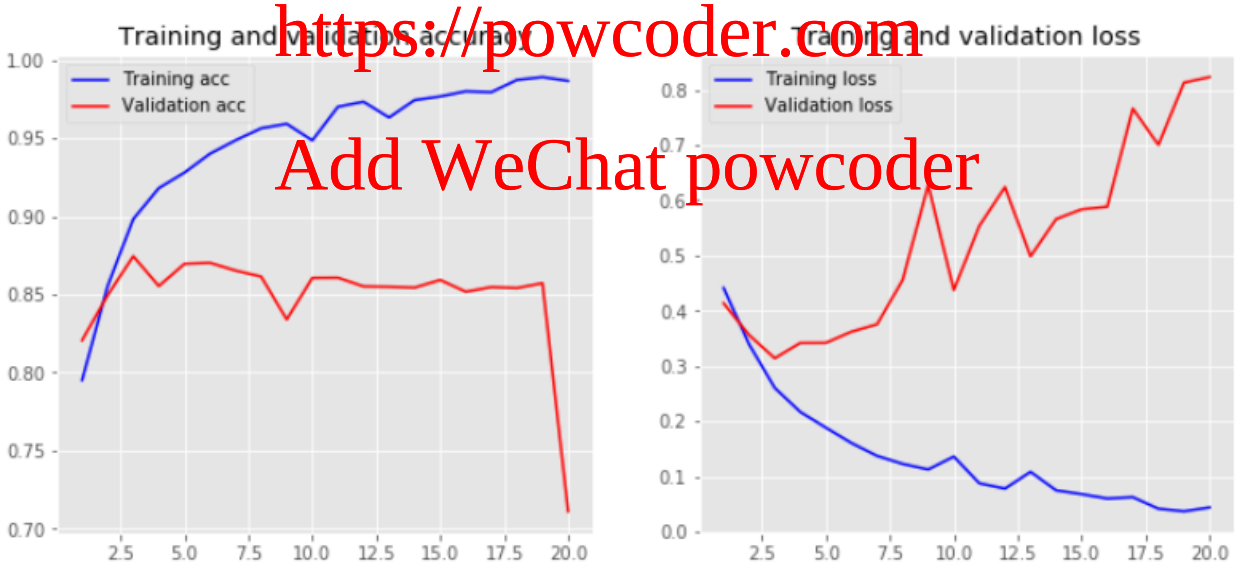
And finally you can evaluate the model using:

```
# Final evaluation of the model
scores = model.evaluate(X_test, y_test, verbose=0)
print("Accuracy: %.2f%%" % (scores[1]*100))
```

Assignment Project Exam Help

Q4- Draw the learning curves and describe them.

https://powcoder.com



Add WeChat powcoder

Q5- Add a dropout to see how the model changes.

Now we want to combine LSTM and CNN. We want to add following code as the convolutional layer.

```
model.add(Conv1D(filters=32, kernel_size=3, padding='same',
activation='relu'))
model.add(MaxPooling1D(pool_size=2))
```

Q6- Add the CNN layer and evaluate the model.

```
Epoch 15/20
391/391 [==============================] - 23s 59ms/step - loss: 0.0184 - accuracy: 0.9959 - val_loss: 0.6426 - val_accuracy: 0.8681
Epoch 16/20
391/391 [==============================] - 23s 59ms/step - loss: 0.0326 - accuracy: 0.9895 - val_loss: 0.6878 - val_accuracy: 0.8647
Epoch 17/20
391/391 [==============================] - 23s 58ms/step - loss: 0.0280 - accuracy: 0.9920 - val_loss: 0.6417 - val_accuracy: 0.8650
Epoch 18/20
391/391 [==============================] - 23s 58ms/step - loss: 0.0108 - accuracy: 0.9982 - val_loss: 0.6879 - val_accuracy: 0.8651
Epoch 19/20
391/391 [==============================] - 23s 58ms/step - loss: 0.0111 - accuracy: 0.9976 - val_loss: 0.6654 - val_accuracy: 0.8630
Epoch 20/20
391/391 [==============================] - 23s 58ms/step - loss: 0.0238 - accuracy: 0.9923 - val_loss: 0.7663 - val_accuracy: 0.8595
Accuracy: 85.95%
```

This is an important aspect of an RNN network that we even have commands for it in Keras.

Simple RNN:

```
keras.layers.SimpleRNN(units, activation='tanh', use_bias=True,
kernel_initializer='glorot_uniform', recurrent_initializer='orthogonal',
bias_initializer='zeros', kernel_regularizer=None,
recurrent_regularizer=None, bias_regularizer=None, activity_regularizer=None,
kernel_constraint=None, recurrent_constraint=None, bias_constraint=None,
dropout=0.0, recurrent_dropout=0.0, return_sequences=False,
return_state=False, go_backwards=False, stateful=False, unroll=False)
```

Assignment Project Exam Help

LSTM:

https://powcoder.com

```
keras.layers.LSTM(units, activation='tanh', recurrent_activation='sigmoid',
use_bias=True, kernel_initializer='glorot_uniform',
recurrent_initializer='orthogonal', bias_initializer='zeros',
unit_forget_bias=True, kernel_regularizer=None, recurrent_regularizer=None,
bias_regularizer=None, activity_regularizer=None, kernel_constraint=None,
recurrent_constraint=None, bias_constraint=None, dropout=0.0,
recurrent_dropout=0.0, implementation=2, return_sequences=False,
return_state=False, go_backwards=False, stateful=False, unroll=False)
```

Add WeChat powcoder

CNN+RNN:

```
keras.layers.ConvLSTM2D(filters, kernel_size, strides=(1, 1),
padding='valid', data_format=None, dilation_rate=(1, 1), activation='tanh',
recurrent_activation='hard_sigmoid', use_bias=True,
kernel_initializer='glorot_uniform', recurrent_initializer='orthogonal',
bias_initializer='zeros', unit_forget_bias=True, kernel_regularizer=None,
recurrent_regularizer=None, bias_regularizer=None, activity_regularizer=None,
kernel_constraint=None, recurrent_constraint=None, bias_constraint=None,
return_sequences=False, go_backwards=False, stateful=False, dropout=0.0,
recurrent_dropout=0.0)
```

Step3. Pytorch- optional

Pytorch is a Python-based scientific computing package targeted at two sets of audiences:

- A replacement for NumPy to use the power of GPUs
- A deep learning research platform that provides maximum flexibility and speed

For installing Pytorch visit:

```
https://pytorch.org/get-started/locally/
```

To test your package you can run:

```
from __future__ import print_function
import torch
x = torch.rand(5, 3)
print(x)
```

Q7- Now write a DNN for MNIST using Pytorch and completely explain your code.

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder