Міністерство освіти та науки України
Львівський національний університет імені Івана Франка

Звіт

Про виконання лабораторної роботи №2

“CNN classification”

Виконав: студент групи ФеС-31 Сало Остап
Перевірив: Сінкевич О.О.

Мета : за допомогою глибокої нейронної мережі CNN навчитися класифікувати зображення.

Замітка: Далі що лежать на основі навчання, я parse власноруч за допомогою бібліотеки google-parse.Я класифікував одяг на людях в 7 категоріях : Воєнні, Поліція, Костюм, Футболка, Бікіні,Плаття,Сорочка.

Реалізація :

Реалізовану було за допомогою нейронної мережі VGG16. framwork tensorflow з використання генераторів для даних.

```
ls
```

```
import pandas as pd
import numpy as np
import itertools
import keras
from sklearn import metrics
from sklearn.metrics import confusion_matrix
from keras.preprocessing.image import ImageDataGenerator, img_to_array, load_img
from keras.models import Sequential
from keras import optimizers
from keras.preprocessing import image
from keras.layers import Dropout, Flatten, Dense
from keras import applications
from keras.utils.np_utils import to_categorical
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
%matplotlib inline
import math
import datetime
import time
from keras.models import Sequential
from keras.layers import Dropout, Flatten, Dense, Activation
from keras.layers.convolutional import Convolution2D, MaxPooling2D
```

Імпорт Бібліотек.

```
#__this can take an hour and half to run so only run it once.
#once the npy files have been created, no need to run again. Convert this cell to a code cell to ru
start = datetime.datetime.now()

generator = datagen.flow_from_directory(
    train_data_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode=None,
    shuffle=False)

nb_train_samples = len(generator.filenames)
num_classes = len(generator.class_indices)

predict_size_train = int(math.ceil(nb_train_samples / batch_size))

bottleneck_features_train = vgg16.predict_generator(generator, predict_size_train)

np.save('clothes_class_train.npy', bottleneck_features_train)
end= datetime.datetime.now()
elapsed= end-start
print ('Time: ', elapsed)
```

Генератори для даних

```
#Default dimensions we found online
img_width, img_height = 600,600

#Create a bottleneck file
top_model_weights_path = 'clothes_clasification.h5'
# loading up our datasets
train_data_dir = 'data/train'
validation_data_dir = 'data/val'
#test_data_dir = 'data/test'

# number of epochs to train top model
epochs = 14 #this has been changed after multiple model run
# batch size used by flow_from_directory and predict_generator
batch_size = 50

nb_filters1 = 32
nb_filters2 = 64
conv1_size = 3
conv2_size = 2
pool_size = 2
classes_num = 7
lr = 0.0001
```

```
#Loading vgc16 model
```

Параметри для навчання

```
#Loading vgc16 model
vgg16 = applications.VGG16(include_top=False, weights='imagenet')
datagen = ImageDataGenerator(rescale=1. / 255)
#needed to create the bottleneck .npy files
```

Зв`язок з VGG16

```
#This is the best model we found. For additional models, check out I_notebook.ipynb
start = datetime.datetime.now()
# model = Sequential()
# model.add(Flatten(input_shape=train_data.shape[1:]))
# model.add(Dense(100, activation=keras.layers.LeakyReLU(alpha=0.3)))
# model.add(Dropout(0.5))
# model.add(Dense(50, activation=keras.layers.LeakyReLU(alpha=0.3)))
# model.add(Dropout(0.3))
# model.add(Dense(num_classes, activation='softmax'))

model = Sequential()
model.add(Convolution2D(nb_filters1, conv1_size, conv1_size, border_mode ="same", input_shape=train_data.shape[1:]))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(pool_size, pool_size)))

model.add(Convolution2D(nb_filters2, conv2_size, conv2_size, border_mode ="same"))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(pool_size, pool_size), dim_ordering='th'))

model.add(Flatten())
model.add(Dense(256))
model.add(Activation("relu"))
model.add(Dropout(0.5))
model.add(Dense(classes_num, activation='softmax'))


model.compile(loss='categorical_crossentropy',
   optimizer=optimizers.RMSprop(lr=1e-4),
   metrics=['acc'])

history = model.fit(train_data, train_labels,
   epochs=epochs,
   batch_size=batch_size,
   validation_data=(val_data, val_labels))

model.save_weights(top_model_weights_path)
```

ting for colab.research.google.com...

Модель .

```
def test_single_image(path_image):
  img_width, img_height = 224, 224
  class_clothes = ['Police','Suit','Bikiny','Mulitary','Shirt','Jaket','Dress']
  print(class_clothes)
  images = read_image(path_image)
  bt_prediction = vgg16.predict(images)
  preds = loaded_model.predict_proba(bt_prediction)
  for idx, class_clothes, x in zip(range(0,6), class_clothes , preds[0]):
    print('ID: {}, Label: {} {}%'.format(idx, class_clothes, round(x*100,2) ))
  print('Final Decision:')
  time.sleep(.5)
  for x in range(3):
    print('.'*(x+1))
    time.sleep(.2)
  class_predicted = loaded_model.predict_classes(bt_prediction)
  #class_dictionary = generator_top.class_indices
  #print('ID: {}, Label: {}'.format(class_predicted[0])
  index = class_predicted[0]
  class_clothes = ['Police','Suit','Bikiny','Mulitary','Shirt','Jaket','Dress']
  print(class_clothes[index])
  display(Image(path_image,width=400, height=400))
```

Функція для результатів

```
[17] path_image = '/content/sample_data/39.jpg'
     test_single_image(path_image)
```

```
['Police', 'Suit', 'Bikiny', 'Mulitary', 'Shirt', 'Jaket', 'Dress']
[INFO] loading and preprocessing image…
ID: 0, Label: Police 0.23%
ID: 1, Label: Suit 0.03%
ID: 2, Label: Bikiny 1.44%
ID: 3, Label: Mulitary 94.19%
ID: 4, Label: Shirt 4.09%
ID: 5, Label: Jaket 0.01%
Final Decision:
.
..
...
Mulitary
```

Результати.

Висновок:

В цій лабораторній роботі я ознайомився з DNN для класифікації зображень.