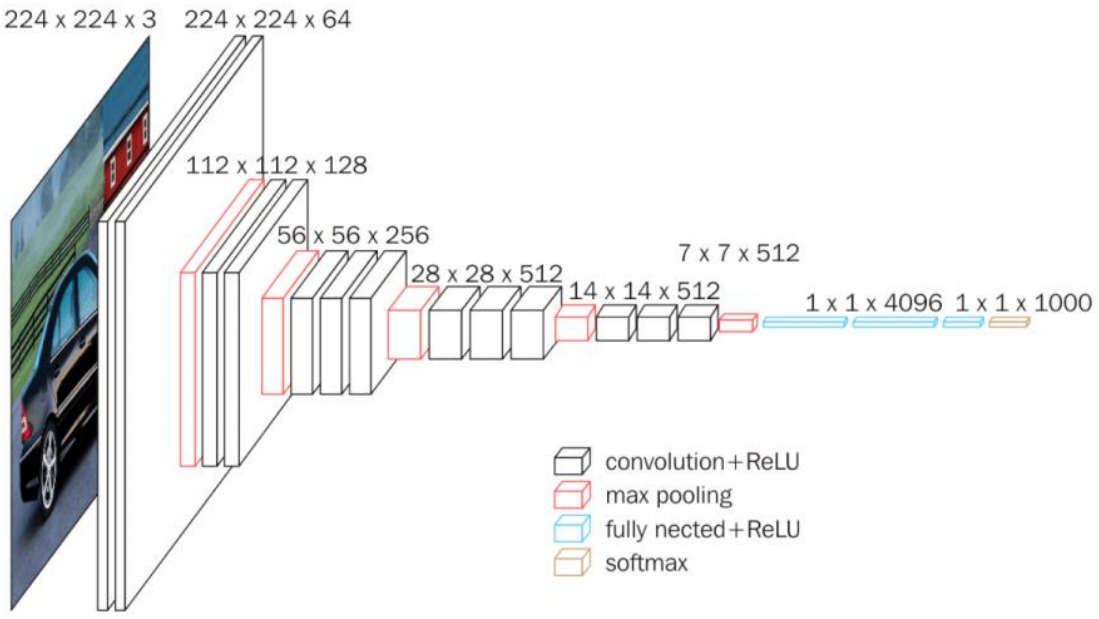
TÌM HIỂU MÔ HÌNH HỌC SÂU
CNN, THƯ VIỆN RESNET,
THƯ VIỆN VGG VÀ FACENET



Sinh viên thực hiện : Trần Đặng Trung Đức

TPHCM, ngày 7 tháng 11 năm 2020

BÀI TẬP

Đề bài Train dữ liệu với 3 mặt người Bill Gate, Mark Zuckerberg, Donald Trump

1. Sử dụng mô hình học sâu CNN của Keras: 3 lớp gồm 1 lớp Convol, 1 lớp Pooling, 1 lớp Fully Connected
2. Sử dụng thư viện Facenet
3. Sử dụng thư viện VGG
4. Sử dụng thư viện Resnet

Bài làm

1. Mô hình CNN

```
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
import matplotlib.pyplot as plt
from keras.preprocessing import image
# Kích thước ảnh
img_height = 110
img_with = 110
batch_size = 10
# Tạo các lớp trong model
model = keras.Sequential([
    layers.Input((110,110,3)),
    layers.Conv2D(16,3,padding='same'),
    layers.MaxPooling2D(pool_size=(2,2)),
    layers.Flatten(),
    layers.Dense(3)]) # Tiền xử lý ảnh cho tập test, tập vali và tập train
ds_train=tf.keras.preprocessing.image_dataset_from_directory(
    'E:/ML and DL/Data/MZ_BG_DT_Train',
    labels="inferred",
    label_mode="int",
    class_names=['BG','DT','MZ'],
    color_mode="rgb",
    batch_size=batch_size,
    image_size=(img_height, img_with),
    shuffle=True,
    seed=123,
    validation_split=0.1,
    subset="training",    )
ds_validation=tf.keras.preprocessing.image_dataset_from_directory(
    'E:/ML and DL/Data/MZ_BG_DT_Train',
    labels="inferred",
    label_mode="int",
    class_names=['BG','DT','MZ'],
    color_mode="rgb",
    batch_size=batch_size,
    image_size=(img_height, img_with),
    shuffle=True,
    seed=123,
    validation_split=0.1,
    subset="validation",    )
```

```
ds_test=tf.keras.preprocessing.image_dataset_from_directory(
    'E:/ML and DL/Data/MZ_BG_DT_Test',
    labels="inferred",
    label_mode="int",
    class_names=['BG','DT','MZ'],
    color_mode="rgb",

    image_size=(img_height, img_with),)

# Làm giàu dữ liệu ảnh
def augment(x,y):
    image = tf.image.random_brightness(x,max_delta=0.05)
    return image,y
ds_train = ds_train.map(augment)

# Compile model và train model
model.compile(
    optimizer = keras.optimizers.Adam(),
    loss = [
        keras.losses.SparseCategoricalCrossentropy(from_logits=True),],
    metrics=["accuracy"],
    )
H = model.fit(ds_train,validation_data=ds_validation, epochs=10,verbose=2)

# Lưu model
model.save('E:/ML and DL/Data/CNN_first_model')
model.save_weights('E:/ML and DL/Data/weight_CNN_first_model.h5')

# Trực quan hóa hàm mất mát và độ chính xác
fig = plt.figure()
numofEpoch = 10
plt.plot(np.arange(0, numofEpoch),H.history['loss'],label='training loss')
plt.plot(np.arange(0, numofEpoch),H.history['val_loss'],label='validation
loss')
plt.plot(np.arange(0, numofEpoch),H.history['accuracy'],label='accuracy')
plt.plot(np.arange(0,
numofEpoch),H.history['val_accuracy'],label='validation accuracy')
plt.title('Accuracy and Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss|Accuracy')
plt.legend()
# Đánh giá model bằng tập test
score = model.evaluate(ds_test, verbose=0)
print(score)
```

```
# Load 3 ảnh trong tập test để dự đoán
img0 = image.load_img('E:/ML and DL/Data/BMD/Bglri 155.png')
img0  = image.img_to_array(img0)
img0  = np.expand_dims(img0, axis=0)
img1 = image.load_img('E:/ML and DL/Data/BMD/Mzlri 151.png')
img1 = image.img_to_array(img1)
img1 = np.expand_dims(img1, axis=0)
img2 = image.load_img('E:/ML and DL/Data/BMD/Dtrrs 148.png')
img2 = image.img_to_array(img2)
img2 = np.expand_dims(img2, axis=0)
img = np.vstack([img0,img1,img2])
img_class=model.predict_classes(img,batch_size=10)
print(img_class)
for things in img_class:
    if(things == [0]):
        print('Bill Gate')
    elif(things == [1]):
        print('Donald Trump')
    else:
        print('Mark Zuckerberg')
# Kết quả chạy code
```

Found 399 files belonging to 3 classes.

Using 360 files for training.

Found 399 files belonging to 3 classes.

Using 39 files for validation.

Found 14 files belonging to 3 classes.

Epoch 1/10

36/36 - 5s - loss: 209.0532 - accuracy: 0.6722 - val_loss: 18.4629 - val_accuracy: 0.7949

Epoch 2/10

36/36 - 1s - loss: 7.1491 - accuracy: 0.9167 - val_loss: 21.8049 - val_accuracy: 0.8205

Epoch 3/10

36/36 - 1s - loss: 2.8586 - accuracy: 0.9639 - val_loss: 29.3157 - val_accuracy: 0.8462

Epoch 4/10

36/36 - 1s - loss: 5.3069 - accuracy: 0.9250 - val_loss: 23.4658 - val_accuracy: 0.8718

Epoch 5/10

36/36 - 1s - loss: 1.3361 - accuracy: 0.9806 - val_loss: 9.4451 - val_accuracy: 0.9231

Epoch 6/10

36/36 - 1s - loss: 0.1901 - accuracy: 0.9944 - val_loss: 13.7092 - val_accuracy: 0.8462

Epoch 7/10

36/36 - 1s - loss: 0.3905 - accuracy: 0.9917 - val_loss: 9.0861 - val_accuracy: 0.8462

Epoch 8/10

36/36 - 1s - loss: 0.7308 - accuracy: 0.9833 - val_loss: 7.3932 - val_accuracy: 0.8974

Epoch 9/10

36/36 - 1s - loss: 0.0068 - accuracy: 0.9972 - val_loss: 10.8417 - val_accuracy: 0.9231

Epoch 10/10

36/36 - 1s - loss: 8.1337e-06 - accuracy: 1.0000 - val_loss: 9.8011 - val_accuracy: 0.9487

[7.399672508239746, 0.9285714030265808]

[0 2 1]

Bill Gate

Mark Zuckerberg

Donald Trump

 # Nhận xét

- **Dự đoán đúng 3 người**
- **Độ chính xác cao trên 90%**

2. Mô hình Facenet

```
from os import listdir
from os.path import isdir
from PIL import Image
from numpy import savez_compressed
from numpy import asarray
from mtcnn.mtcnn import MTCNN
import numpy as np
from keras.models import load_model
from numpy import expand_dims
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import Normalizer
from sklearn.svm import SVC
# Hàm trích xuất ảnh
def extract_face(filename, required_size=(160, 160)):
        # Load ảnh từ file
        image = Image.open(filename)
        # Convert ảnh thành RBG nếu cần
        image = image.convert('RGB')
        # Convert thành mảng
        pixels = asarray(image)
        # Tạo detector
        detector = MTCNN()
        # Phát hiện khuôn mặt trong ảnh
        results = detector.detect_faces(pixels)
        # Trích xuất bounding box
        x1, y1, width, height = results[0]['box']
        # Gỡ lỗi
        x1, y1 = abs(x1), abs(y1)
        x2, y2 = x1 + width, y1 + height
        # Trích xuất khuôn mặt
        face = pixels[y1:y2, x1:x2]
        # Resize lại kích thước phù hợp với model
        image = Image.fromarray(face)
        image = image.resize(required_size)
        face_array = asarray(image)
        return face_array
```

```
# Hàm load khuôn mặt
def load_faces(directory):
	faces = list()
	for filename in listdir(directory):
		# Đường dẫn
		path = directory + filename
		# Lấy khuôn mặt
		face = extract_face(path)
		# Lưu vào mảng faces
		faces.append(face)
	return faces
# Hàm load dữ liệu ảnh
def load_dataset(directory):
	X, y = list(), list()
	for subdir in listdir(directory):
		# Đường dẫn
		path = directory + subdir + '/'
		# Skip file trong thư mục chính
		if not isdir(path):
			continue
		# Load tất cả ảnh  trong thư mục con
		faces = load_faces(path)
		# Tạo label
		labels = [subdir for _ in range(len(faces))]
		# In kết quả load
		print('>loaded %d examples for class: %s' % (len(faces),subdir))
		# Lưu trong X và y
		X.extend(faces)
		y.extend(labels)
	return asarray(X), asarray(y)
# Load dữ liệu train
trainX, trainy = load_dataset('E:/ML and DL/Data/Facenet/MZ_BG_DT_Train/')
print(trainX.shape, trainy.shape)
# Load dữ liệu test
testX, testy = load_dataset('E:/ML and DL/Data/Facenet/MZ_BG_DT_Validation/')
# Lưu thành một file nén
savez_compressed('E:/ML and DL/Data/MZ_DT_BG.npz', trainX, trainy, testX, testy)
# Hàm lấy ảnh nhúng
def get_embedding(model, face_pixels):
	# Scale giá trị pixel
	face_pixels = face_pixels.astype('float32')
	# Chuẩn hóa
	mean, std = face_pixels.mean(), face_pixels.std()
	face_pixels = (face_pixels - mean) / std
	# Transform ảnh thành mẫu
	samples = expand_dims(face_pixels, axis=0)
	# Dự đoán để lấy mẫu nhúng
	yhat = model.predict(samples)
	return yhat[0]
```

```
# Load file npz
data = np.load('E:/ML and DL/Data/MZ_BZ_DT.npz')
trainX, trainy, testX, testy = data['arr_0'], data['arr_1'], data['arr_2'],
data['arr_3']
print('Loaded: ', trainX.shape, trainy.shape, testX.shape, testy.shape)
# Load model
model = load_model('E:/ML and DL/Data/facenet_keras.h5')
print('Loaded Model')
# Convert mỗi ảnh trong tập train thành nhúng
newTrainX = list()
for face_pixels in trainX:
        embedding = get_embedding(model, face_pixels)
        newTrainX.append(embedding)
newTrainX = asarray(newTrainX)
print(newTrainX.shape)
# Convert mỗi ảnh trong tập test thành nhúng
newTestX = list()
for face_pixels in testX:
        embedding = get_embedding(model, face_pixels)
        newTestX.append(embedding)
newTestX = asarray(newTestX)
print(newTestX.shape)
print('Dataset: train=%d, test=%d' % (trainX.shape[0], testX.shape[0]))
# Chuẩn hóa vector đầu vào
in_encoder = Normalizer(norm='l2')
trainX = in_encoder.transform(trainX)
testX = in_encoder.transform(testX)
# Mã hóa nhãn
out_encoder = LabelEncoder()
out_encoder.fit(trainy)
trainy = out_encoder.transform(trainy)
testy = out_encoder.transform(testy)
# Fit model
model = SVC(kernel='linear', probability=True)
model.fit(trainX, trainy)
# Dự đoán
yhat_train = model.predict(trainX)
yhat_test = model.predict(testX)
# Đánh giá
score_train = accuracy_score(trainy, yhat_train)
score_test = accuracy_score(testy, yhat_test)
# In kết quả
print('Accuracy: train=%.3f, test=%.3f' % (score_train*100, score_test*100))
```

 # Kết quả khi chạy code

>loaded 105 examples for class: BG

>loaded 131 examples for class: DT

>loaded 134 examples for class: MZ

(370, 160, 160, 3) (370,)

>loaded 29 examples for class: BG

>loaded 28 examples for class: DT

>loaded 27 examples for class: MZ

Loaded: (370, 160, 160, 3) (370,) (84, 160, 160, 3) (84,)

Loaded Model

(370, 128)

(84, 128)

Dataset: train=370, test=84

Accuracy: train=100.000, test=100.000

 # Nhận xét

- **Độ chính xác rất cao**

3. Mô hình VGG

```
import numpy as np
from tensorflow import keras
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from keras.layers import Dense, Flatten, Dropout, BatchNormalization
from keras.models import Sequential
from keras.applications.vgg16 import VGG16
from keras.preprocessing import image
import matplotlib.pyplot as plt
# Khai báo các kích thước và đường dẫn
Image_size = [110,110]
train_path = 'E:/ML and DL/Data/MZ_BG_DT_Train'
valid_path = 'E:/ML and DL/Data/MZ_BG_DT_Validation'
img_height = 110
img_with = 110
batch_size = 10
# Tạo dữ liệu hình ảnh
datagen = ImageDataGenerator(rescale=1./255)
train_generator = datagen.flow_from_directory(directory=train_path,
                                              target_size=(img_height,img_with),
                                              classes=['BG','DT','MZ'],
                                              class_mode='categorical',
                                              batch_size=batch_size)
validation_generator = datagen.flow_from_directory(directory=valid_path,

target_size=(img_height,img_with),
                                                   classes=['BG','DT','MZ'],
                                                   class_mode='categorical',
                                                   batch_size=batch_size)
# Tiền xử lý dùng model VGG16
vgg = VGG16(input_shape=(img_height,img_with,3), weights='imagenet',
include_top=False)
# Không train với layer trong vgg
for layer in vgg.layers:
    layer.trainable = False
# Tạo thêm các layer cho model và fit model
model = Sequential()
model.add(vgg)
model.add(Flatten())
model.add(Dense(128,activation='relu'))
model.add(Dropout(0.2))
model.add(BatchNormalization())
model.add(Dense(1,activation="sigmoid"))
model.compile(optimizer="adam",
              loss=[keras.losses.CategoricalCrossentropy(from_logits=True),],
              metrics=["accuracy"],
              )
```

```
H = model.fit(train_generator,validation_data=validation_generator,
epochs=10,verbose=2)
# Lưu model
model.save('E:/ML and DL/Data/VGG16_first_model')
model=keras.models.load_model('E:/ML and DL/Data/VGG16_first_model')
# Load ảnh trong tập test
img0 = image.load_img('E:/ML and DL/Data/BMD/Dtrrs 148.png')
img0  = image.img_to_array(img0)
img0 = img0/255.0
img0  = np.expand_dims(img0, axis=0)
img0_class=np.argmax(model.predict(img0),axis=1)
print(img0_class)
# Kết quả khi chạy code
```

Found 399 images belonging to 3 classes.

Found 84 images belonging to 3 classes.

Epoch 1/10

40/40 - 17s - loss: 1.0986 - accuracy: 0.5013 - val_loss: 1.0986 - val_accuracy: 0.5357

Epoch 2/10

40/40 - 14s - loss: 1.0986 - accuracy: 0.5071 - val_loss: 1.0986 - val_accuracy: 0.5476

Epoch 3/10

40/40 - 13s - loss: 1.0986 - accuracy: 0.5021 - val_loss: 1.0986 - val_accuracy: 0.5437

Epoch 4/10

40/40 - 13s - loss: 1.0986 - accuracy: 0.5071 - val_loss: 1.0986 - val_accuracy: 0.5159

Epoch 5/10

40/40 - 13s - loss: 1.0986 - accuracy: 0.5096 - val_loss: 1.0986 - val_accuracy: 0.4643

Epoch 6/10

40/40 - 13s - loss: 1.0986 - accuracy: 0.4962 - val_loss: 1.0986 - val_accuracy: 0.4921

Epoch 7/10

40/40 - 13s - loss: 1.0986 - accuracy: 0.4937 - val_loss: 1.0986 - val_accuracy: 0.4643

Epoch 8/10

40/40 - 13s - loss: 1.0986 - accuracy: 0.4879 - val_loss: 1.0986 - val_accuracy: 0.4762

```
Epoch 9/10

40/40 - 13s - loss: 1.0986 - accuracy: 0.4946 - val_loss: 1.0986 - val_accuracy:
0.4722

Epoch 10/10

40/40 - 13s - loss: 1.0986 - accuracy: 0.4879 - val_loss: 1.0986 - val_accuracy:
0.4841

[0]

# Nhận xét
   -  Độ chính xác khá thấp, không lớn hơn 50%
```

4. Mô hình Resnet

```
import numpy as np
from tensorflow import keras
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from keras.layers import Dense, Flatten, Dropout, BatchNormalization
from keras.models import Sequential
from keras.applications.resnet import ResNet50
from keras.preprocessing import image
import matplotlib.pyplot as plt
# Khai báo các kích thước và đường dẫn
Image_size = [110,110]
train_path = 'E:/ML and DL/Data/MZ_BG_DT_Train'
valid_path = 'E:/ML and DL/Data/MZ_BG_DT_Validation'
img_height = 110
img_with = 110
batch_size = 10
# Tạo dữ liệu hình ảnh
datagen = ImageDataGenerator(rescale=1./255)
train_generator = datagen.flow_from_directory(directory=train_path,
                                              target_size=(img_height,img_with),
                                              classes=['BG','DT','MZ'],
                                              class_mode='categorical',
                                              batch_size=batch_size)
validation_generator = datagen.flow_from_directory(directory=valid_path,

target_size=(img_height,img_with),
                                                   classes=['BG','DT','MZ'],
                                                   class_mode='categorical',
                                                   batch_size=batch_size)
```

```
# Tiền xử lý dùng model Resnet50
resnet = ResNet50(input_shape=(img_height,img_with,3), weights='imagenet',
include_top=False)
# Không train với layer trong resnet
for layer in resnet.layers:
    layer.trainable = False
# Tạo thêm các layer cho model và fit model
model = Sequential()
model.add(resnet)
model.add(Flatten())
model.add(Dense(128,activation='relu'))
model.add(Dropout(0.2))
model.add(BatchNormalization())
model.add(Dense(1,activation="sigmoid"))
model.compile(optimizer="adam",
              loss=[keras.losses.CategoricalCrossentropy(from_logits=True),],
              metrics=["accuracy"],
              )
H = model.fit(train_generator,validation_data=validation_generator,
epochs=10,verbose=2)
# Lưu model
model.save('E:/ML and DL/Data/ResNet50_first_model')
# Load model
model=keras.models.load_model('E:/ML and DL/Data/ResNet50_first_model')
# Load ảnh trong tập test
img0 = image.load_img('E:/ML and DL/Data/BMD/Dtrrs 148.png')
img0  = image.img_to_array(img0)
img0 = img0/255.0
img0  = np.expand_dims(img0, axis=0)
img0_class=np.argmax(model.predict(img0),axis=1)
print(img0_class)
```

Found 399 images belonging to 3 classes.

Found 84 images belonging to 3 classes.

Epoch 1/10

40/40 - 9s - loss: 1.0986 - accuracy: 0.4979 - val_loss: 1.0986 - val_accuracy: 0.6667

Epoch 2/10

40/40 - 8s - loss: 1.0986 - accuracy: 0.5063 - val_loss: 1.0986 - val_accuracy: 0.6667

Epoch 3/10

40/40 - 8s - loss: 1.0986 - accuracy: 0.4971 - val_loss: 1.0986 - val_accuracy: 0.6587

Epoch 4/10

40/40 - 8s - loss: 1.0986 - accuracy: 0.4854 - val_loss: 1.0986 - val_accuracy: 0.5794

Epoch 5/10

40/40 - 8s - loss: 1.0986 - accuracy: 0.4862 - val_loss: 1.0986 - val_accuracy: 0.6071

Epoch 6/10

40/40 - 8s - loss: 1.0986 - accuracy: 0.4879 - val_loss: 1.0986 - val_accuracy: 0.5754

Epoch 7/10

40/40 - 8s - loss: 1.0986 - accuracy: 0.4904 - val_loss: 1.0986 - val_accuracy: 0.5198

Epoch 8/10

40/40 - 8s - loss: 1.0986 - accuracy: 0.5129 - val_loss: 1.0986 - val_accuracy: 0.5556

Epoch 9/10

40/40 - 8s - loss: 1.0986 - accuracy: 0.5297 - val_loss: 1.0986 - val_accuracy: 0.4802

Epoch 10/10

40/40 - 8s - loss: 1.0986 - accuracy: 0.5414 - val_loss: 1.0986 - val_accuracy: 0.4960

[0]

 # Nhận xét

- **Độ chính xác khá thấp, khoảng 54%**