Deep Learning for Business Development

Celebrity image classification using CNN

Submitted To

Dr. Revendranath Tirumalsety

(Department of Management)

Department of Management

BITS-Pilani, Pilani Campus

2021-2023

Date: 18-11-2022

Submitted By:

Vishva Bhalodiya [ID:2021H1540833P]

Mitta Naveen Kumar Reddy [ID:2021H1540843P]

Sudhanshu Ranjan[ID:2021H154024P]

Kartik Pandey [ID:2021H1540844P]

Introduction:

This project demonstrates the classification of Bollywood celebrities using the concept of the convolution neural network in deep learning. The Convolutional Neural Network (CNN or ConvNet) is mainly used for image and speech recognition applications. Its built-in convolutional layer reduces the high dimensionality of images without losing their information. That is why CNN are especially suited for this use case.

Motivation & Analysis:

Have you ever wondered which celebrity looks similar to you? If yes, then this project will be exciting for you. The project includes the classification of celebrities and will also help identify the celebrity who looks similar to you.

Businesses and governments can further use this concept to profile their customers based on their images, or a highly advanced application of this concept would be to check if the food went stale just on the basis of its image.

We used images of celebrities as they are readily available online, which helped in training our model easily.

Outline of the approach:

1. Importing required libraries & package
2. Data loading
3. Data Augmentation
4. Data Pre-Preparation
5. Model Building & prediction

1. Required Library and Packages

pandas, NumPy, matplotlib.pyplot, math, keras.preprocessing.image import ImageDataGenerator

tensorflow.keras.utils, glob,cv2,tqdm,tflearn,tensorflow,os

2. Data loading

Dataset: 10 celebrity images with around 80 to 150 images each.

3. Data Augmentation

We augmented the data so that our neurons can be trained in a better manner as it can identify different variations of the same image, hence when a test image is run in the model, our model can give output with better accuracy.

```
datagen = ImageDataGenerator(
        rotation_range=45,
        width_shift_range=0.2,
        height_shift_range=0.2,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True,
        fill_mode='nearest')
```

4. Data Pre-Preparation

The data was labeled using label_img function and prepared using the create_train_data function.

```
def create_train_data():
   training_data = []
   for img in training_files:
        print(img)
        label = label_img(img)
        img = cv2.imread(img,
cv2.IMREAD_GRAYSCALE)
        img = cv2.resize(img,
(IMG_SIZE, IMG_SIZE))

training_data.append([np.array(img)
, np.array(label)])
   shuffle(training_data)
   np.save('train_data.npy',
training_data)
   return training_data
```

```
def label_img(img):
    word_label =
path.normpath(img).split('.')[0].s
plit('\\')[-2]
    index=labels.index(word_label)
    output=np.zeros(len(labels),
dtype = int)
    output[index]=1
    return output
```

5. Model Building & prediction

Hyperparameters of model:

learning rate: LR = 1e-3 (Learning rate shouldn't be too high or low) this is good

No of epoch= 30

```
from tensorflow.keras import datasets, layers, models

tf.compat.v1.reset_default_graph()

model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu',
input_shape=(IMG_SIZE, IMG_SIZE, 1)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(32, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dense(1024, activation='relu'))
model.add(layers.Dense(len(labels), activation = 'sigmoid'))
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=1e-3),
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 48, 48, 32)	320
max_pooling2d (MaxPooling2D)	(None, 24, 24, 32)	0
conv2d_1 (Conv2D)	(None, 22, 22, 64)	18496
max_pooling2d_1 (MaxPooling 2D)	(None, 11, 11, 64)	0
conv2d_2 (Conv2D)	(None, 9, 9, 64)	36928
max_pooling2d_2 (MaxPooling 2D)	(None, 4, 4, 64)	0
conv2d_3 (Conv2D)	(None, 2, 2, 32)	18464
max_pooling2d_3 (MaxPooling 2D)	(None, 1, 1, 32)	0
flatten (Flatten)	(None, 32)	0
dense (Dense)	(None, 1024)	33792

dense_1 (Dense) **(None, 10)** **10250**

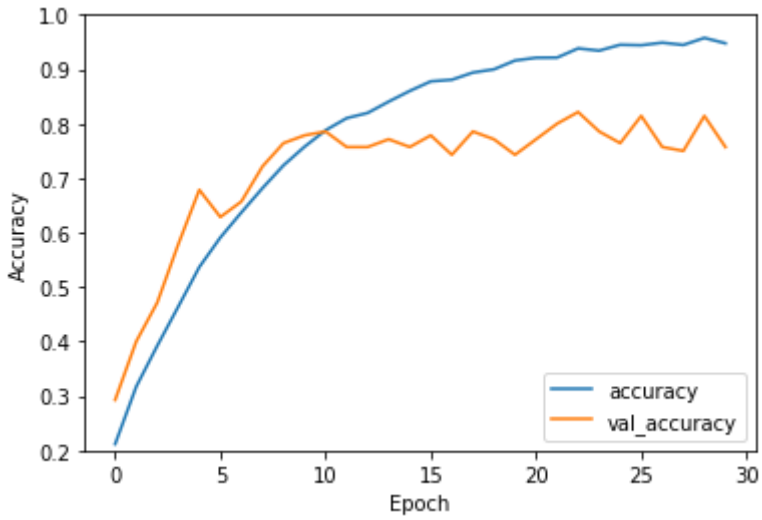
Total params: 118,250
Trainable params: 118,250
Non-trainable params: 0
_____

```
history = model.fit(X,Y,epochs=30,
                    validation_data=(val_x,val_y))
```

Output:

Epoch 30/30
14069/14069 [==============================] - 16s 1ms/sample - loss: 0.1451 - acc: 0.9476 - val_loss: 1.2418 - val_acc: 0.7571

Prediction output:



Accuracy vs Epochs

Results with test data

As we can see our model has accurately predicted the below 2 celebrities.


```
[1.0000000e+00 2.2665506e-17 9.1465537e-11 9.5722300e-01 2.4504396e-14
 3.6580888e-05 5.1535512e-06 9.4969835e-08 5.2190703e-06 1.2123311e-11]
This image most likely belongs to Aishwarya_Rai with a 100.00 percent confidence.
```


```
[3.6502934e-05 1.0000000e+00 2.0084410e-06 1.5404455e-15 9.9998003e-01
 1.6903361e-07 1.0922512e-09 5.4933333e-01 1.0332703e-06 3.7167331e-03]
This image most likely belongs to Ajay_Devgn with a 100.00 percent confidence.
```

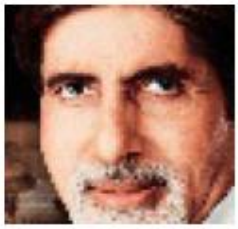
Results with real data

With little modification, our model can also tell the similarity between a person and the celebrity's images which have been trained in our model.

```
predict_imag("Faces\MBA Class/6.png")
```

```
This image most likely belongs to Ayushmann_Khurrana with a 99.99 percent confidence.
This image most likely belongs to Amitabh_Bachchan with a 99.26 percent confidence.
This image most likely belongs to Disha_Patani with a 1.08 percent confidence.
```






```
predict_imag("Faces\MBA Class/7.png")
```

```
This image most likely belongs to Ayushmann_Khurrana with a 100.00 percent confidence.
This image most likely belongs to Amitabh_Bachchan with a 5.69 percent confidence.
This image most likely belongs to Disha_Patani with a 0.14 percent confidence.
```




```
This image most likely belongs to Ayushmann_Khurrana with a 100.00 percent confidence.
This image most likely belongs to Hrithik_Roshan with a 99.83 percent confidence.
This image most likely belongs to Emraan_Hashmi with a 81.96 percent confidence.
```