

Proctored Examination System

Deploying Neural network model for Human detection using flask in the server end

Description :

The concept is to make a examination system for Online application, where it can be invigilated using AI (Deep learning model) for any kind of malpractice. In the web application system the web-cam of the client system will take pictures with a certain interval of time. Those pictures will be sent to the backend server for verification and detection if the pictures contains a human facing forward on his/her computer screen or not. It can detect things such as :

1. No presence of human.
2. Fake Image of a human.
3. Can classify between real Human from Human like objects.

Required software and libraries :

Here we have used python for our neural network model and also used libraries – Tensorflow, Keras, Scikit-Image, Numpy, ImageDataGenerator.

Dataset :

We have used a dataset “Human vs Non human” . Where there are pictures of human faces and other category is for non human objects like cat, dog, car, plane etc.

Model making & training :

The model has been created with multiple layers using Convolution and max-pooling, and dropping off some nodes with poor performance. It has been trained for 10 epochs resulting in a validation accuracy of about 94%.

Deployment :

For the actual system, we have created a html page with some javascript on it to click pictures of the user and sending it to the backend. In the backend we have flask where the neural network model will be preloaded for prediction of Human presence in the pictures.

Procedure :

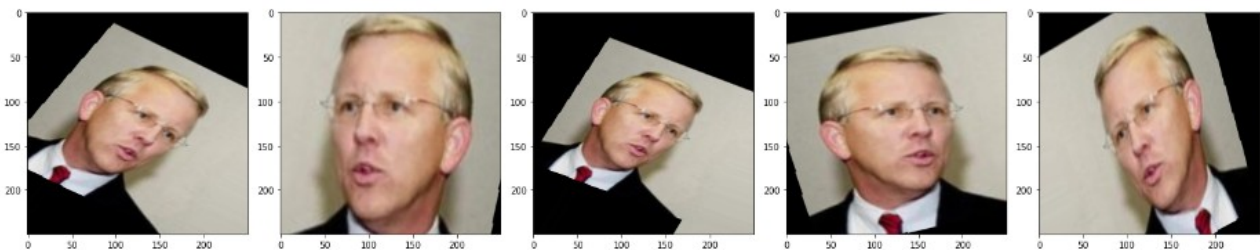
1. The Source code for neural network model has been developed in google colab, as It is easier to find libs available.
2. The dataset has been uploaded to a google drive and afterwards imported to google colab, and accordingly extracted.

```
from google.colab import drive
drive.mount('/content/gdrive',force_remount=True)
```

Mounted at /content/gdrive

```
!unzip /content/gdrive/MyDrive/archive.zip
```

3. Now for better results with different types of human position in front of camera, the dataset images has been augmented using ImageDataGenerator from keras.
Eg : Rotation, shearing, zoom etc



4. The Model has been created using keras 2D convolutional layer with maxpooling for input layer and has a single hidden layer of 512 neurons , and one output layer with two alternatives.

Eg:

```
model=tf.keras.models.Sequential(layers=[
    tf.keras.layers.Conv2D(filters=32,padding='same',kernel_size=(3,3),activation='relu',input_shape=(Img_shape,Img_shape,3)),
    tf.keras.layers.MaxPooling2D(2),
    tf.keras.layers.Conv2D(filters=64,padding='same',kernel_size=(3,3),activation='relu'),
    tf.keras.layers.MaxPooling2D(2),
    tf.keras.layers.Conv2D(filters=128,padding='same',kernel_size=(3,3),activation='relu'),
    tf.keras.layers.MaxPooling2D(2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(units=512,activation='softmax'),
    tf.keras.layers.Dense(units=2)
])
```

5. Finally compiling and training the model for 10 epochs.

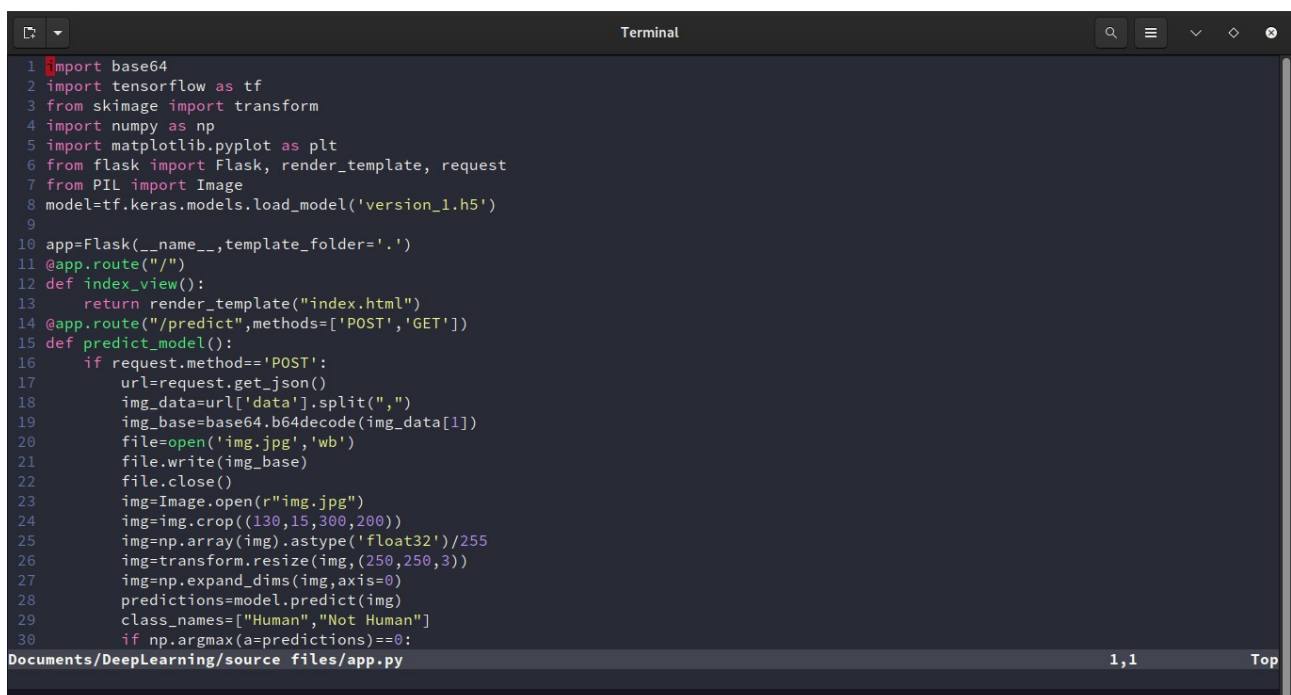
```
Epoch 1/10
41/41 [=====] - 125s 3s/step - loss: 0.6196 - accuracy: 0.8360 - val_loss: 0.5766 - val_accuracy: 0.9486
Epoch 2/10
41/41 [=====] - 115s 3s/step - loss: 0.5547 - accuracy: 0.9543 - val_loss: 0.5636 - val_accuracy: 0.8814
Epoch 3/10
41/41 [=====] - 115s 3s/step - loss: 0.5225 - accuracy: 0.9469 - val_loss: 0.5099 - val_accuracy: 0.9423
Epoch 4/10
41/41 [=====] - 114s 3s/step - loss: 0.4871 - accuracy: 0.9607 - val_loss: 0.5021 - val_accuracy: 0.9067
Epoch 5/10
41/41 [=====] - 114s 3s/step - loss: 0.4777 - accuracy: 0.9311 - val_loss: 0.4903 - val_accuracy: 0.8939
Epoch 6/10
41/41 [=====] - 114s 3s/step - loss: 0.4400 - accuracy: 0.9558 - val_loss: 0.5017 - val_accuracy: 0.8542
Epoch 7/10
41/41 [=====] - 115s 3s/step - loss: 0.4216 - accuracy: 0.9516 - val_loss: 0.4055 - val_accuracy: 0.9607
Epoch 8/10
41/41 [=====] - 116s 3s/step - loss: 0.3948 - accuracy: 0.9615 - val_loss: 0.4167 - val_accuracy: 0.9251
Epoch 9/10
41/41 [=====] - 116s 3s/step - loss: 0.3735 - accuracy: 0.9632 - val_loss: 0.3800 - val_accuracy: 0.9475
Epoch 10/10
41/41 [=====] - 115s 3s/step - loss: 0.3560 - accuracy: 0.9647 - val_loss: 0.3802 - val_accuracy: 0.9321
```

For practical uses the model will be saved for deployment in a backend server, where requests from client will be served.

Proctored Examination System

Start Test

For backend application flask has been used :



```
1 import base64
2 import tensorflow as tf
3 from skimage import transform
4 import numpy as np
5 import matplotlib.pyplot as plt
6 from flask import Flask, render_template, request
7 from PIL import Image
8 model=tf.keras.models.load_model('version_1.h5')
9
10 app=Flask(__name__,template_folder='.')
11 @app.route("/")
12 def index_view():
13     return render_template("index.html")
14 @app.route("/predict",methods=['POST','GET'])
15 def predict_model():
16     if request.method=='POST':
17         url=request.get_json()
18         img_data=url['data'].split(",")
19         img_base=base64.b64decode(img_data[1])
20         file=open('img.jpg','wb')
21         file.write(img_base)
22         file.close()
23         img=Image.open(r"img.jpg")
24         img=img.crop((130,15,300,200))
25         img=np.array(img).astype('float32')/255
26         img=transform.resize(img,(250,250,3))
27         img=np.expand_dims(img,axis=0)
28         predictions=model.predict(img)
29         class_names=["Human","Not Human"]
30         if np.argmax(a=predictions)==0:
```

Documents/DeepLearning/source files/app.py 1,1 Top

End of Document