Liveness Detection API

The API constructed is based on the liveness detection of a person standing in front of a surveillance camera, i.e whether a real person is standing or it's just an image that is held by some other person.

IMPLEMENTATION STEPS -

- The model is implemented based on the **number of times the person has blinked** using **pre-trained face detector and facial landmarks detector**. It uses something called the **eye aspect ratio**(EAR) whose value goes down rapidly when the person blinks and that is detected by the model.
- A threshold value is defined below which if the EAR goes then the model detects that a blink has occurred, and another threshold value that should match the no of consecutive frames(in a live video stream) for which one blink has occurred.
- An infinite loop is used to initiate the live video stream, then on each iteration - the frames are captured, the faces are detected for each frame captured, the facial landmarks are plotted around the eyes of the detected faces, EAR is monitored constantly and **if a blink is detected then it's a real person**.

API SPECIFICATIONS -

- The API is constructed using the **Django REST Framework**
- It is based on **token based authentication**, i.e a person has to **first give his/her username and password** to register. The **token** received after registering can be used to access the API.
- *Input* - a **video of a person blinking** .
  Output - **the name, liveness**(whether real or fake) **and the no of blinks** made by that person **as a json response**.
  NOTE : This API works on a recorded video and not on a live video stream.
- The API is hosted using the local django server through the url - https://127.0.0.1:8000/file/eye_blink
  (The API is tested using the *postman app*)

STEPS TO RUN -

- The **user should register** himself/herself which will authenticate that user and provide him/her the **token**(through postman only) required to access the API as defined in the api specs part.

Reference - https://medium.com/quick-code/token-based-authentication-for-django-rest-framework-44586a9a56fb

- Then the **input** as defined above in the api specifications part should be **provided through request** to the API.(i.e a video of a person blinking visibly)
- As an output, the json response will be displayed defined in the **output** section above.

COLAB NOTEBOOK

https://colab.research.google.com/drive/1IZ5LfKBzgNSvXWAzvh4t0XTKZMNB3Ga9?usp=sharing

REQUIREMENTS -

- Before testing and running the api, one should make sure the following packages and libraries should be installed in their machine
    1. Django(version >= 2.2.4)
    2. Django Rest Framework(version >= 3.0.0)
    3. open cv
    4. dlib
    5. imutils
    6. face_recognition
    7. numpy
    8. scipy
- Implement the code with **csrf exemption**, send the **required inputs through requests** including the **auth token**.
  NOTE : The complete testing part is done in the **POST** method
- The **pretrained models** include -
    1. **haarcascade_frontalface_alt.xml** -> detects the face locations in each captured frame and records the coordinates in a tuple of the form (x, y, w, h). Where (x, y) is the centre coordinate and h is height & w is weight.
    2. **mmod_human_face_detector.dat** -> similar to haarcascade_frontalface_alt.xml
    3. **shape_predictor_68_face_landmarks.dat** -> detects and records the coordinates of the facial features of a face such as jaw, eyes, nose, lips, etc.
- The IDE used for the implementation and execution of the above can be carried out with open sources like
    1. Pycharm(community version)
    2. Sublime Text
    3. PyDev