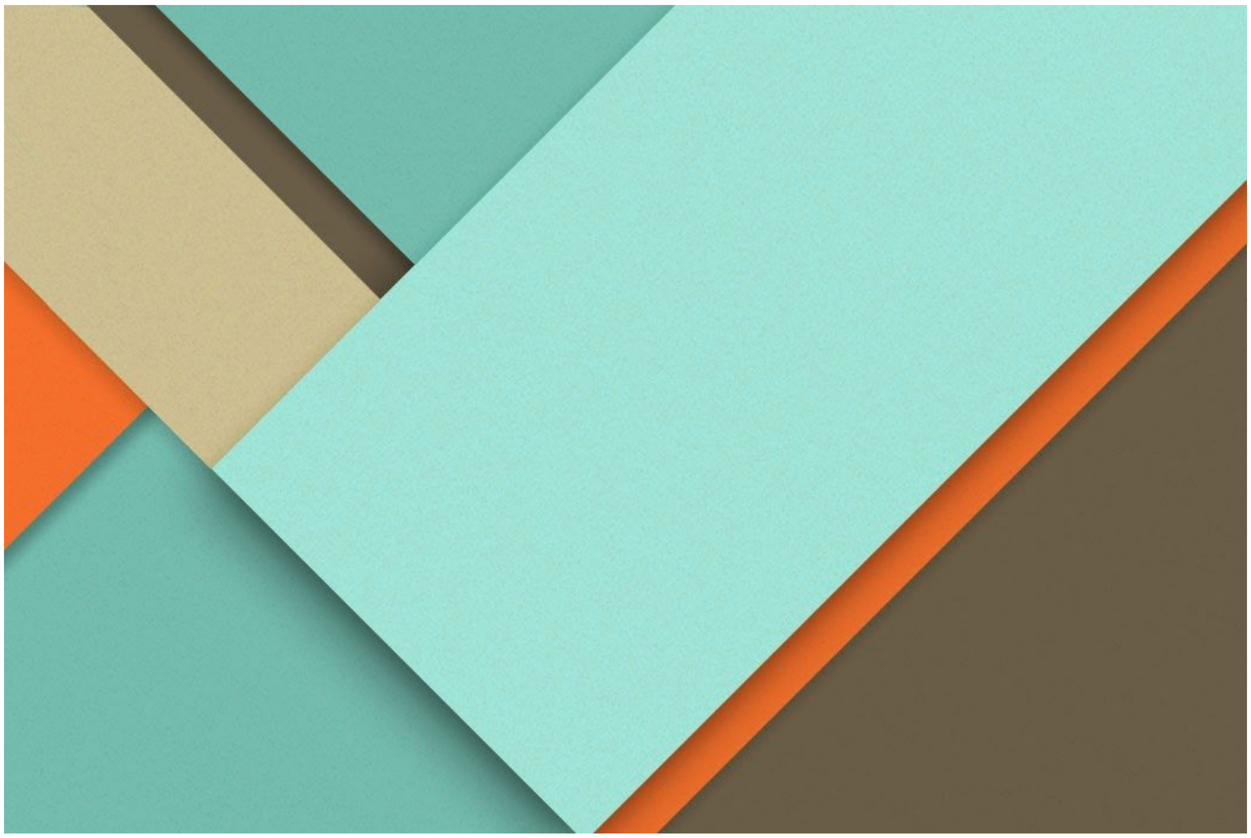


Liveness Detection

09.07.2019

Arunaav
arunaav@iiitb.org
INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY BANGALORE

INTRODUCTION:

Most facial recognition algorithms you find on the internet and research papers suffer from photo attacks. These methods work really well at detecting and recognizing faces in images, videos and video streams from webcam. However they can't distinguish between real life faces and faces on a photo. This inability to recognize faces is due to the fact that these algorithms work on 2D frames.In this project I made an attempt to detect real and fake faces.

PACKAGES USED:

Keras

Opencv

Dlib

PIL

Filterpy

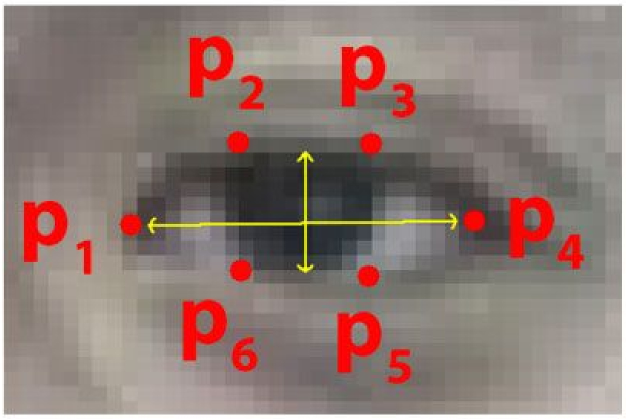
Torch

PROPOSED METHOD:

The proposed method is composed of two steps.In the first step we'll detect the eye blinks and in the second step the trained classifier predicts whether the face is real or fake.

Each detected face in the frame is tracked using SORT (Simple Online and Realtime Tracking) ,this algorithm can track multiple objects in real time but the algorithm merely associates already detected objects across different frames based on the coordinates of detection results .SORT uses mathematical heuristics such as maximizing the IOU (intersection-over-union) metrics between bounding boxes in neighboring frames. Each box is labeled with a number (object id), and if there is no relevant box in the next frame, the algorithm assumes that the object has left the frame.

Now each face with a specific id will undergo a blink detection test .For blink test I used '*eye aspect ratio*' to determine if a person is blinking or not in a given video frame.To find eye aspect ratio we need facial landmarks of both the eyes and i used dlib's face landmark detector to get coordinates of both the eyes. Each eye is represented by 6 *(x, y)*-coordinates, starting at the left-corner of the eye (as if you were looking at the person), and then working clockwise around the remainder of the region.

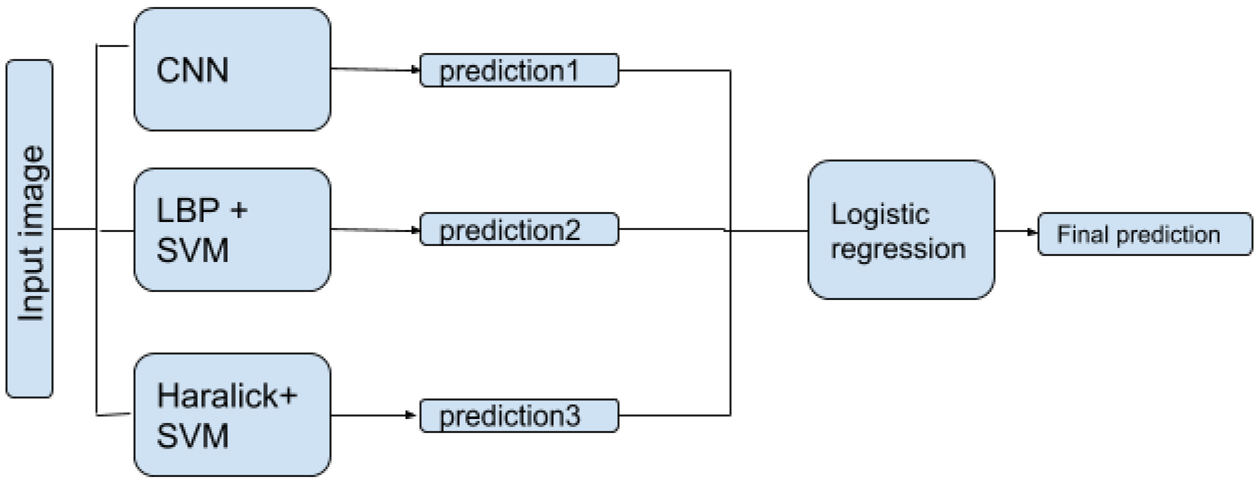


Based on the co-ordinates ,we can calculate eye aspect ratio using the formula given below:

$$\text{EAR} = \frac{|p_2 - p_6| + |p_3 - p_5|}{2|p_1 - p_4|}$$

The numerator of this equation computes the distance between the vertical eye landmarks while the denominator computes the distance between horizontal eye landmarks, weighting the denominator appropriately since there is only *one* set of horizontal points but *two* sets of vertical points.If the EAR value goes rapidly below a certain threshold value (the distance between the vertical eye landmarks is reduced i.e. eye is closed) and increases within a certain number of frames which indicates a blink.

In the second step the pre-trained classifier will predict if the detected face is fake or real.I used texture feature LBP(Local Binary Patterns) and spatial feature Haralick feature vectors to train the non-linear SVM model and I also trained a Deep learning CNN on the same dataset and stacked the predictions of both the models to make the final predictions.The pipeline of training process is given below.



CNN:

A multi-layered convolutional neural network is used for liveness detection.The Structure of the CNN model is depicted in the figure shown below.

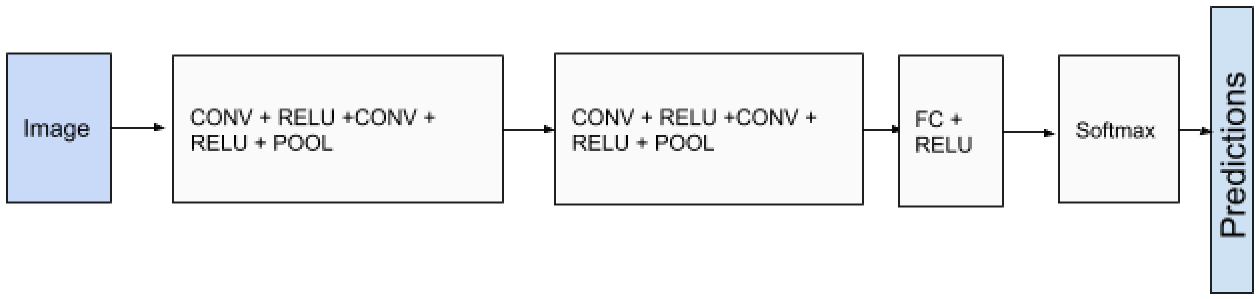
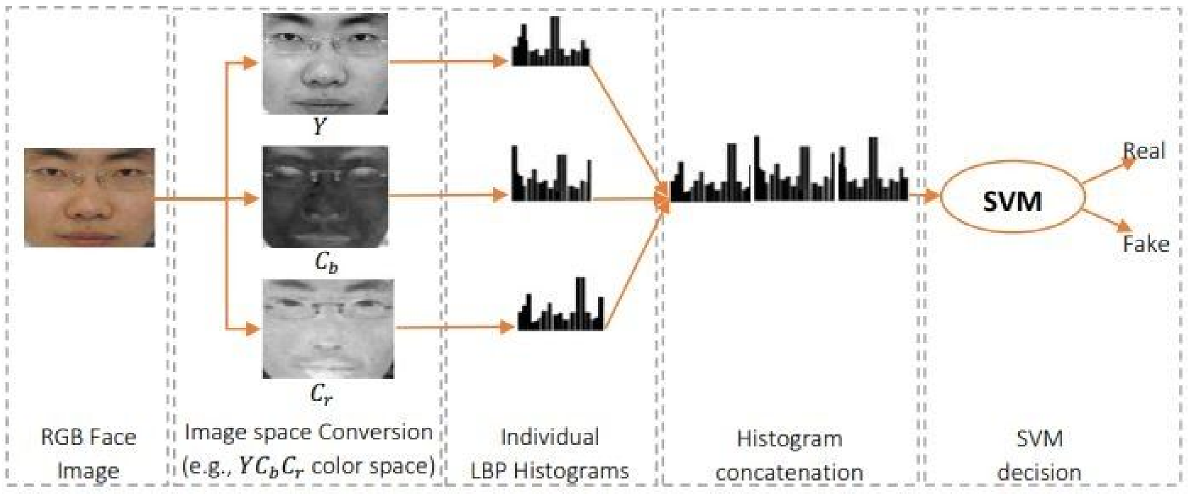


Image
CONV + RELU +CONV + RELU + POOL
CONV + RELU +CONV + RELU + POOL
FC + RELU
Softmax
Predictions

LBP+SVM:

To detect spoofing attacks, the color LBP features are extracted from face images are fed into a Support Vector Machine (SVM) classifier. The general diagram of our proposed approach for detecting spoofing attacks is shown below.

For LBP feature extraction of we convert the image from RGB to YCrCb and we'll LBP on each separate color field and concatenate the individual histograms .This concatenated histogram is sent to SVM classifier for decision making.


Y
C_b
C_r
SVM
Real
Fake
RGB Face Image
Image space Conversion (e.g., YC_bC_r color space)
Individual LBP Histograms
Histogram concatenation
SVM decision

Folder Structure:

Liveness Detection:

```
|    gather_examples.py    (used to create dataset by extracting images from videos)
|    test.py               (main file used to conduct face liveness detection in real-time)
|    train.py              (this will train the model and save it in the same folder)
|    le.pickle             ( class label encoder )
|    liveness.model        (serialized Keras model which detects face liveness.)
|    livenessnet.py        (CNN model)
|    Lr.pickle
|    shape_predictor_68_face_landmarks.dat
|    SVM1.pickle
|    SVM2.pickle
|
├──────data                    (NUAA Dataset for Liveness detection)
|    ├──────fake
|    └──────real
|
├──────data2                   (Generated Dataset)
|    ├──────fake
|    └──────real
|
├──────face_detector           (Used to detect the faces in a given frame)
|      deploy.prototxt
|      res10_300x300_ssd_iter_140000.caffemodel
|
└──────utils
     |   models.py
     |   sort.py
     ├──────utils
```

train.py:

This python script will load the data and will extract the texture features from the faces ,it uses LBP and haralick features to train a SVM model and a CNN is also trained with the same dataset to detect fake and real faces .Once the training is finished ,it will save all the trained models for real-time testing.

test.py :

This script will load all the pre trained models and will turn on the webcam to grab the frames . It uses a pre trained Caffe face detector to detect the face in each frame and each detection tracked by SORT algorithm and it assigns a unique id to each face in frame.we'll extract feature from detected faces and predict if it is real or fake.

gather_examples.py:

This script grabs face ROIs from input video files and helps us to create a dataset.

Acknowledgements:

This project is done under the guidance and mentorship of G. Shobha Rani .

References:

- https://arxiv.org/pdf/1511.06316.pdf
- https://www.pyimagesearch.com/2019/03/11/liveness-detection-with-opencv/
- https://www.pyimagesearch.com/2015/12/07/local-binary-patterns-with-python-opencv/
- https://github.com/imironica/liveness
- https://ieeexplore.ieee.org/document/7012923
- https://towardsdatascience.com/real-time-face-liveness-detection-with-python-keras-and-opencv-c35dc70dafd3