

Greatlearning, Chennai
University of Texas at Austin
McCombs School of Business

Home security system with Intruder Alert using Computer Vision

A project submitted in fulfillment of the requirements of
the PG Program in Artificial Intelligence & Machine Learning

Sivakumar Panneerselvam
Anton Erone Pius
Hariharan Ramesh
Jayaseeli Joachin
Arun P

November 2020

Acknowledgments

We express our sincere gratitude to Dr. Narayana, Professor, AIML, GreatLearning, for his continuous supervision, feedback and helpful suggestions at various stages of the project.

We wish to extend our thanks to R. Karthik, our mentor, for his help, guidance and insightful comments to improve the quality of this project.

We are extremely thankful to the faculty team of AIML, Aug'19 batch, for equipping us with the skills necessary to tackle this project.

Table of Contents

1. Summary of Problem statement
2. Overview of the final process
3. Solution walk-through
4. Architecture
5. Model
6. Comparison with Benchmark
7. Visualisations
8. Implications
9. Limitations
10. Conclusion
11. Literature survey
12. References

Abstract

Providing reliable security for homes in the absence of a house owner is crucial nowadays. In this project, we present a solution for home security systems, using Computer Vision. We use OpenCV library to implement Face detection and Face recognition algorithms in python.

The implemented system depends on the SSD framework on ResNet and CNN. It recognizes faces captured on the security camera installed at home, in a shorter time with better accuracy.

This system can scale well when new training data are uploaded often to authenticate known reliable faces.

Summary of Problem statement

Provision of home security services has become an integral part of our lives in today's technological society where attackers usually have all the necessary means and resources at their disposal. The traditional home security system, i.e. closed circuit television (CCTV) can only capture and record the video without being able to give warning feedback if there is any suspicious/unauthorized human activities within the perimeter.

In light of this problem we have developed a unified intruder alert system with face recognition. The face detection module differentiates between authorized people and intruders. With the video footage of the authorized personnel, using OpenCV and Tensorflow the model is trained to identify all the authorized personnel.

Caffemodel network is used to identify the position of the face from the input image frames. We train the model with different facial features such as beard, spectacles etc., to make sure the authorized person aren't classified as intruders, in addition the fake face/face liveness detection is implemented to identify the intruder from the authorized personnel's face mask/picture. Once the intruder is identified an alert message is triggered to all the authorized personals.

Overview of the final process

To address the problem at hand, it is imperative that the model is trained with the facial embeddings of the personalities the model is trying to identify.

When programming a face recognition problem we need to identify the faces in the given video feed or the images that are being processed, further that the model needs to be able to classify the face that is being identified.

Solving Methodology:

The solution begins with identifying all the users who have clearance for entering the given secured space (eg, home, office buildings). The identified users are then taken a video of 10 to 15 secs in all the possible angles. These videos will serve as the training dataset for the model. The videos taken will have all kinds of background and in this case we need to consider them as noise.

The video is processed to convert them into individual frames, these frames are then further processed to get the face embeddings from the individual frames. The retrieval of face embeddings will enable us to get the faces of the authorized users separated from noises of the frame. The retrieved face embeddings are then saved as individual pictures in a folder, which is later supplied as a training dataset to the tensorflow model.

The facial embedding is retrieved using a caffe model - **res10_300x300_ssd_iter_140000.caffemodel**, along with this model there is a predefined weights that is available, using which the face embeddings are retrieved and stored locally. With this the training dataset preparation and preprocessing is completed.

The model defined in this project is a simple tensorflow model which has a total params: **148,822**. The model is designed as a light-weight model so that it can be deployed in a raspberry pi system.

Using the open-cv module we parse the video feed through the model to identify and recognise the face. The model can predict the faces with **90+% accuracy**.

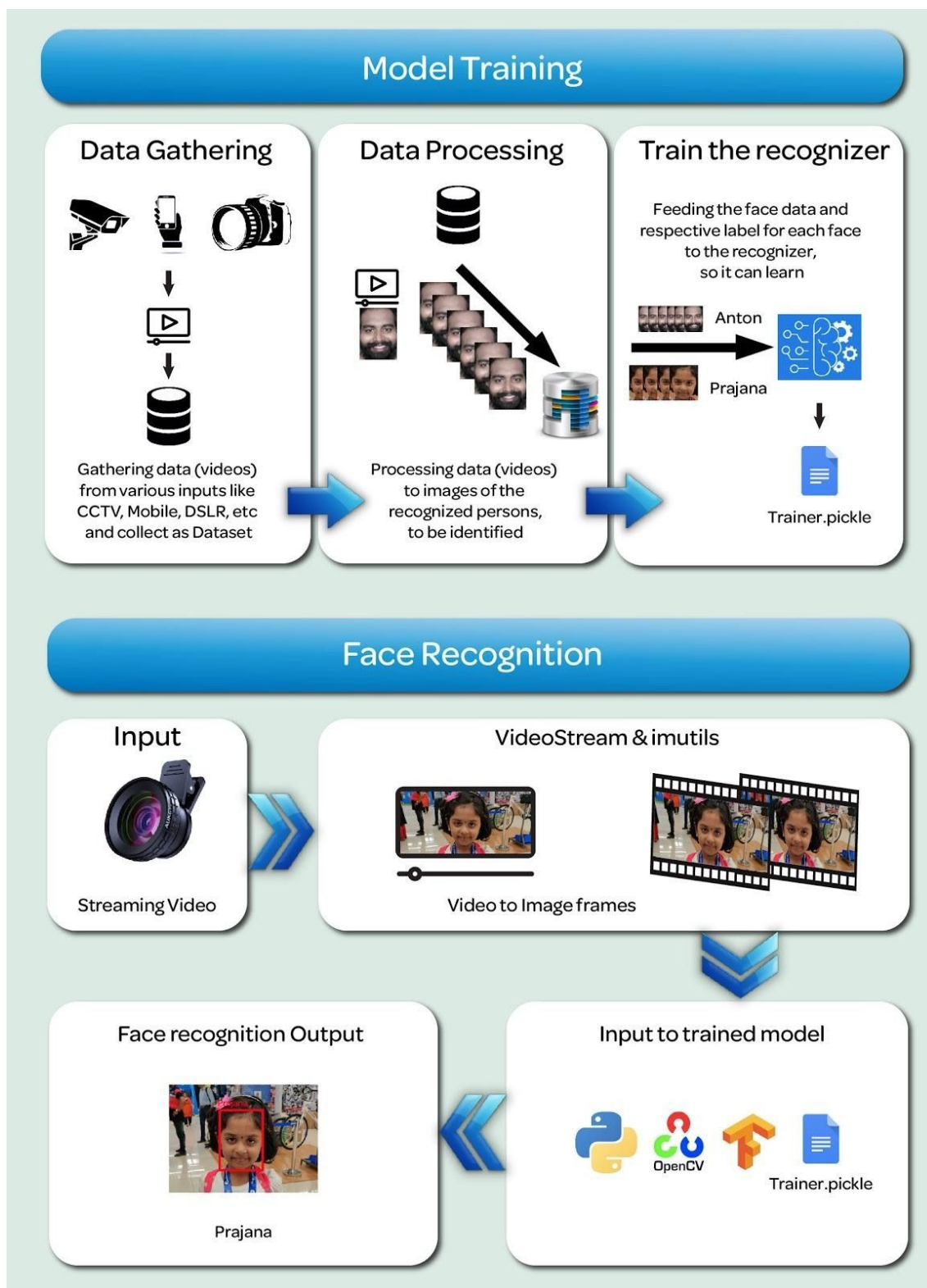
This entire project is broken into simple .py scripts and can be invoked from the cmd line. From the raw video conversion to frames, from frames to face embeddings, from training the model and saving the trained model as a .h5 file and then running a live cam for face recognition and identification, all the scripts are controlled in a cmd line.

Solution walk-through

The solution is a simple tensorflow model of size **1.81MB**. This model can be deployed on any remote system that can use the video feed to predict the faces in the video frame.

1. The live video feed is converted to frames.
2. The obtained frames are further processed to isolate the face embeddings from the frame's noise.
3. The face embeddings are then resized to **32*32*3 image size**.
4. The resized face embeddings retrieved from the frame is then given as an input feature of the model.
5. The model then gives a prediction with an accuracy of who might be the person on the camera feed.

Architecture



Model

Consider what would happen if a nefarious user tried to purposely circumvent your face recognition system.

Such a user could try to hold up a photo of another person. Maybe they even have a photo or video on their smartphone that they could hold up to the camera responsible for performing face recognition (such as in the image at the top of this post).

In those situations it's entirely possible for the face held up to the camera to be correctly recognized...but ultimately leading to an unauthorized user bypassing face recognition system!

How would you go about spotting these “fake” versus “real/legitimate” faces? How could you apply anti-face spoofing algorithms into your facial recognition applications?

The answer is to apply **liveness detection** with OpenCV

LivenessNet is the final model is being used for this project to apply liveness detection

To create our liveness detection we utilized OpenCV, Deep Learning, and Python.

The first step was to gather our real vs. fake dataset. To accomplish this task, we:

1. First recorded a video of ourselves using our smartphone (i.e., “real” faces).
2. Held our smartphone up to our laptop/desktop, replayed the same video, and then recorded the replaying using our webcam (i.e., “fake” faces).
3. Applied face detection to both sets of videos to form our final liveness detection dataset.

After building our dataset we implemented, “LivenessNet”, a Keras + Deep Learning CNN.

This network is purposely shallow, ensuring that:

1. We reduce the chances of overfitting on our small dataset.
2. The model itself is capable of running in real-time (including on the Raspberry Pi).

Overall, our liveness detector was able to obtain 90+% accuracy on our validation set.

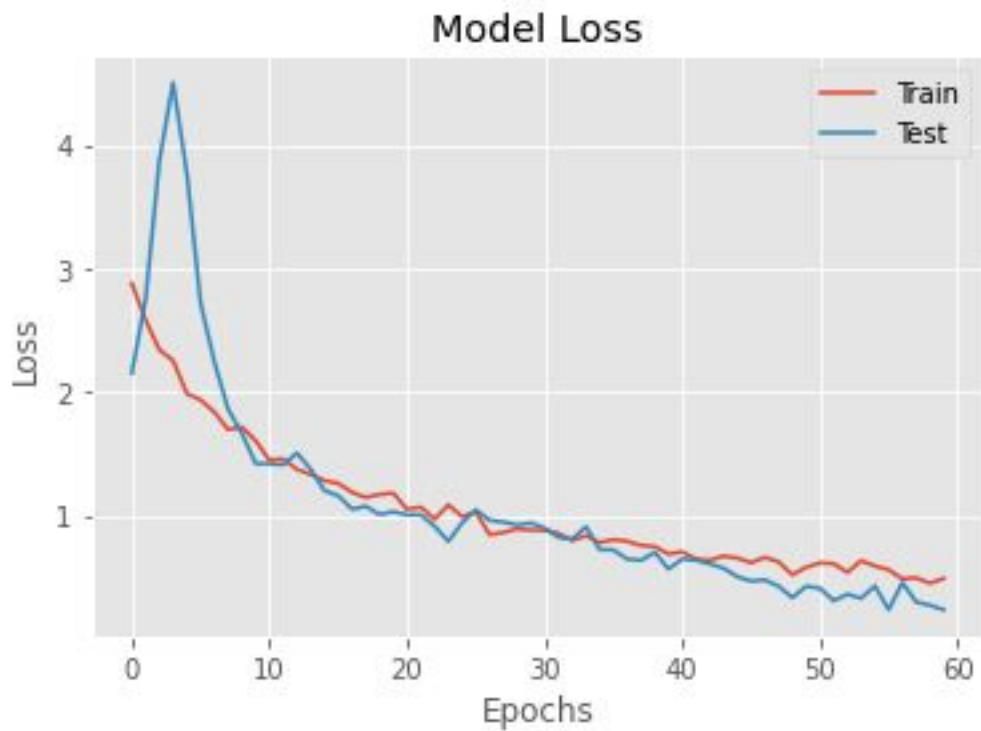
Model Summary

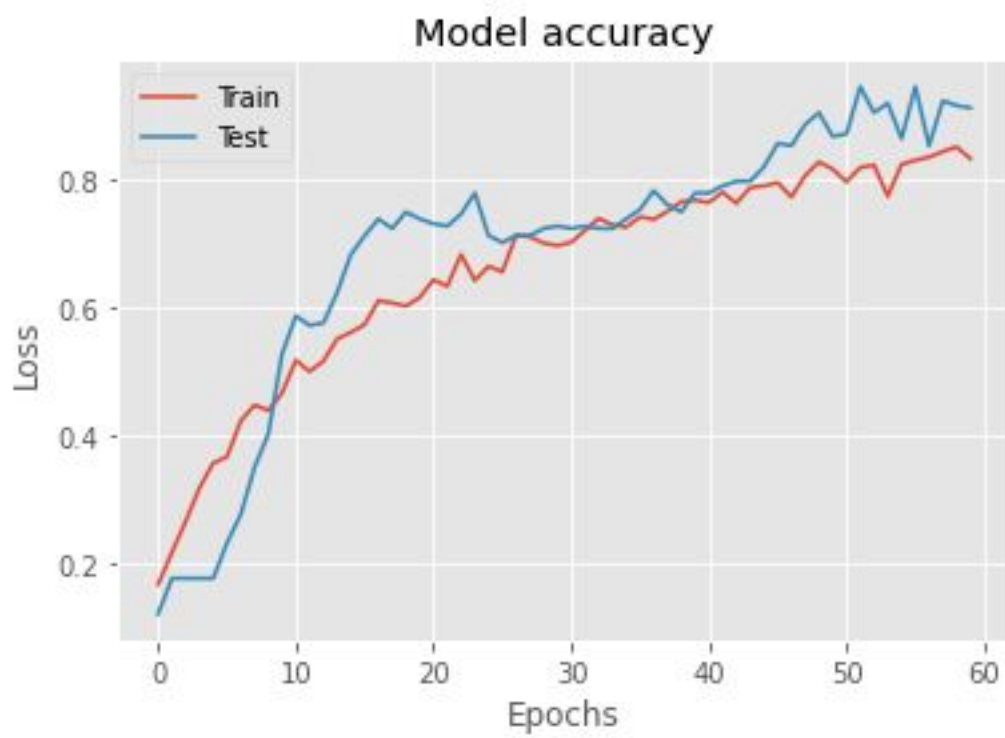
Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 32, 32, 16)	448
activation (Activation)	(None, 32, 32, 16)	0
batch_normalization (Batch Normalization)	(None, 32, 32, 16)	64
conv2d_1 (Conv2D)	(None, 32, 32, 16)	2320
activation_1 (Activation)	(None, 32, 32, 16)	0
batch_normalization_1 (Batch Normalization)	(None, 32, 32, 16)	64
max_pooling2d (MaxPooling2D)	(None, 16, 16, 16)	0
dropout (Dropout)	(None, 16, 16, 16)	0
conv2d_2 (Conv2D)	(None, 16, 16, 32)	4640
activation_2 (Activation)	(None, 16, 16, 32)	0
batch_normalization_2 (Batch Normalization)	(None, 16, 16, 32)	128
conv2d_3 (Conv2D)	(None, 16, 16, 32)	9248
activation_3 (Activation)	(None, 16, 16, 32)	0
batch_normalization_3 (Batch Normalization)	(None, 16, 16, 32)	128
max_pooling2d_1 (MaxPooling2D)	(None, 8, 8, 32)	0
dropout_1 (Dropout)	(None, 8, 8, 32)	0
conv2d_4 (Conv2D)	(None, 8, 8, 32)	9248
activation_4 (Activation)	(None, 8, 8, 32)	0
batch_normalization_4 (Batch Normalization)	(None, 8, 8, 32)	128
conv2d_5 (Conv2D)	(None, 8, 8, 32)	9248
activation_5 (Activation)	(None, 8, 8, 32)	0
batch_normalization_5 (Batch Normalization)	(None, 8, 8, 32)	128
max_pooling2d_2 (MaxPooling2D)	(None, 4, 4, 32)	0
dropout_2 (Dropout)	(None, 4, 4, 32)	0
conv2d_6 (Conv2D)	(None, 4, 4, 32)	9248
activation_6 (Activation)	(None, 4, 4, 32)	0
batch_normalization_6 (Batch Normalization)	(None, 4, 4, 32)	128
conv2d_7 (Conv2D)	(None, 4, 4, 32)	9248
activation_7 (Activation)	(None, 4, 4, 32)	0
batch_normalization_7 (Batch Normalization)	(None, 4, 4, 32)	128

max_pooling2d_3 (MaxPooling2)	(None, 2, 2, 32)	0
dropout_3 (Dropout)	(None, 2, 2, 32)	0
flatten (Flatten)	(None, 128)	0
dense (Dense)	(None, 64)	8256
activation_8 (Activation)	(None, 64)	0
batch_normalization_8 (Batch Normalization)	(None, 64)	256
dropout_4 (Dropout)	(None, 64)	0
dense_1 (Dense)	(None, 7)	455
activation_9 (Activation)	(None, 7)	0
=====		
Total params: 63,511		
Trainable params: 62,935		
Non-trainable params: 576		

Model Success





Comparison with Benchmark

Due to the nature of the business problem, the model performance is largely based on the dataset collected specifically for the Recognition purpose of known faces. Hence no direct comparisons were made with any existing solutions.

The objective of the project is achieved using this pipeline :

1. Face Detection,
2. Model Training for Feature Extraction and
3. Face Recognition at real time, for verification / Authentication.

The Face Detection is an important task in the Face Recognition process. The Face Detection process will be accurate, when the detection model is trained with a large dataset of facial images. Due to this specific requirement of the business problem, it was decided to use a pre-trained Face Detector model that is trained with obvious facial features, which is shipped with **OpenCV**.

This model is based on Single Shot Detector (**SSD**) framework, with a ResNet backbone network that is based on VGG, which is defined and trained using the Caffe Deep Learning Framework. The model consists of the following two files :

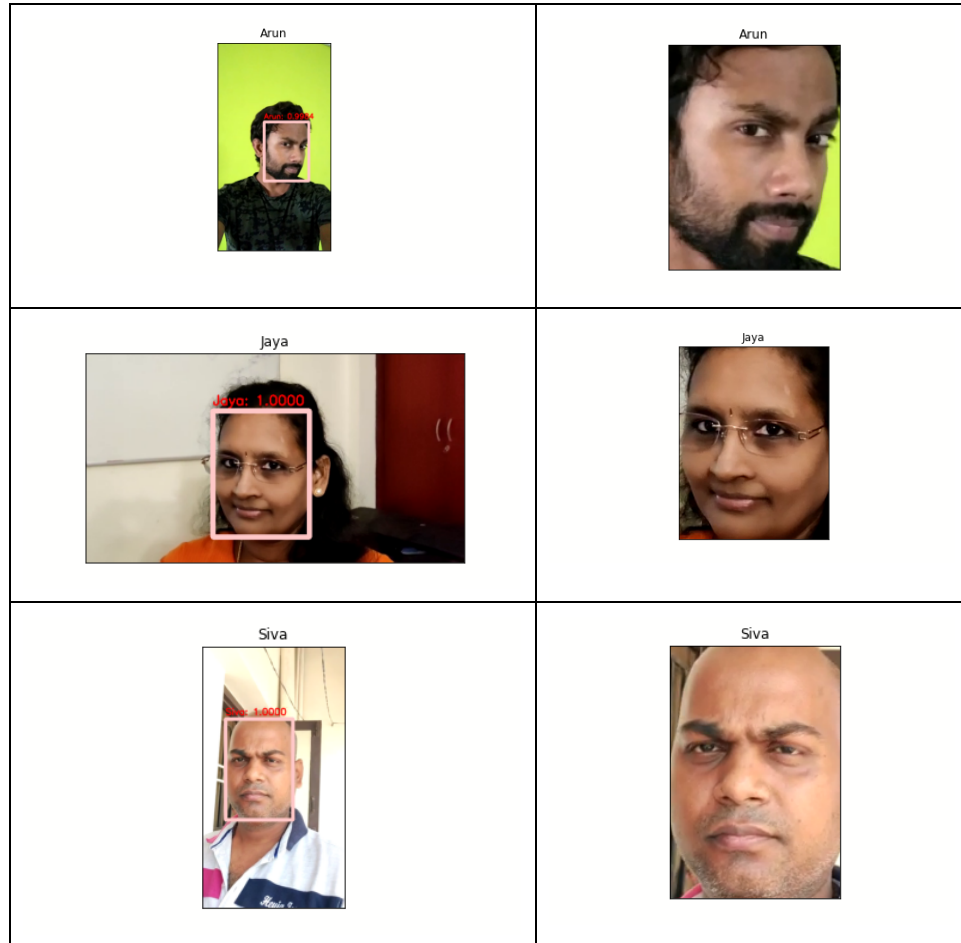
- The network definition (**deploy.prototxt**)
- The learned weights (**res10_300x300_ssd_iter_140000.caffemodel**)



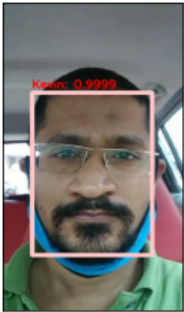
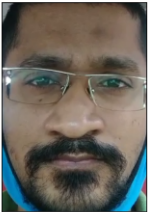
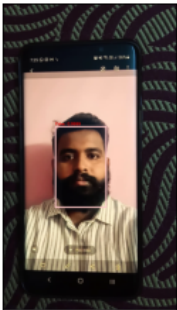

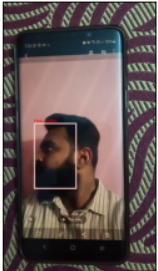

The model used **WIDER FACE** dataset, a benchmark dataset, which was proposed in **CVPR 2016**. The dataset has 32,203 images on 61 event classes and 393,703 faces with bounding box annotations with the format of “[left, top, width, height]”. This dataset contains faces with large variations in expression, pose, illumination and has many tiny faces too.

Visualisations

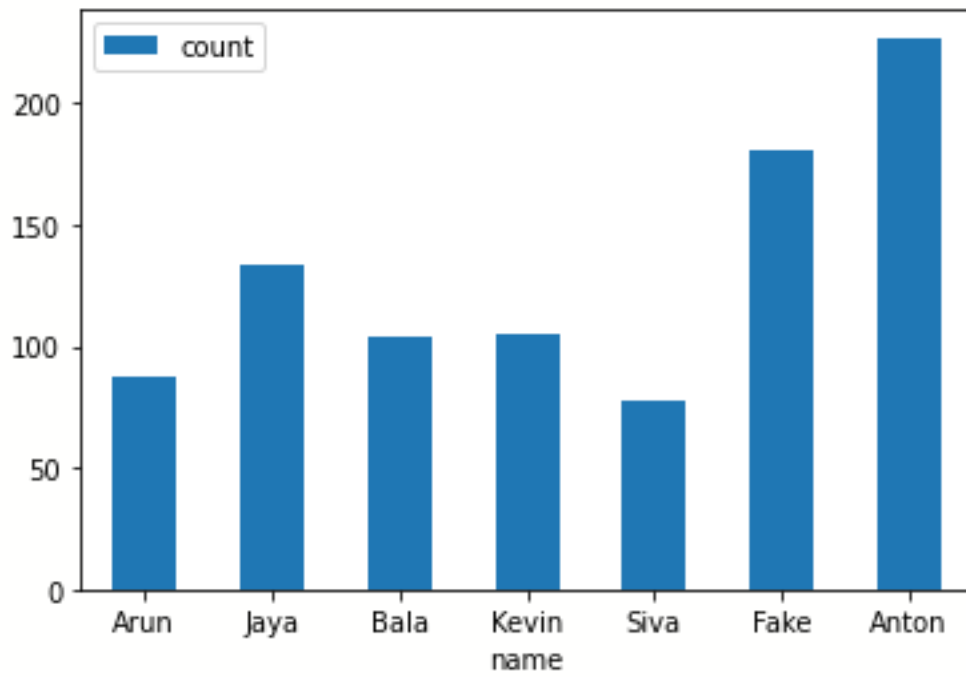
The following are samples of the extracted face ROI bounding boxes from gathered face datasets, after Face Detection.

The Face Detector model detects, crops and extracts face ROIs. Each face ROI is written to a separate file, which forms a dataset for training Face Recognition model.

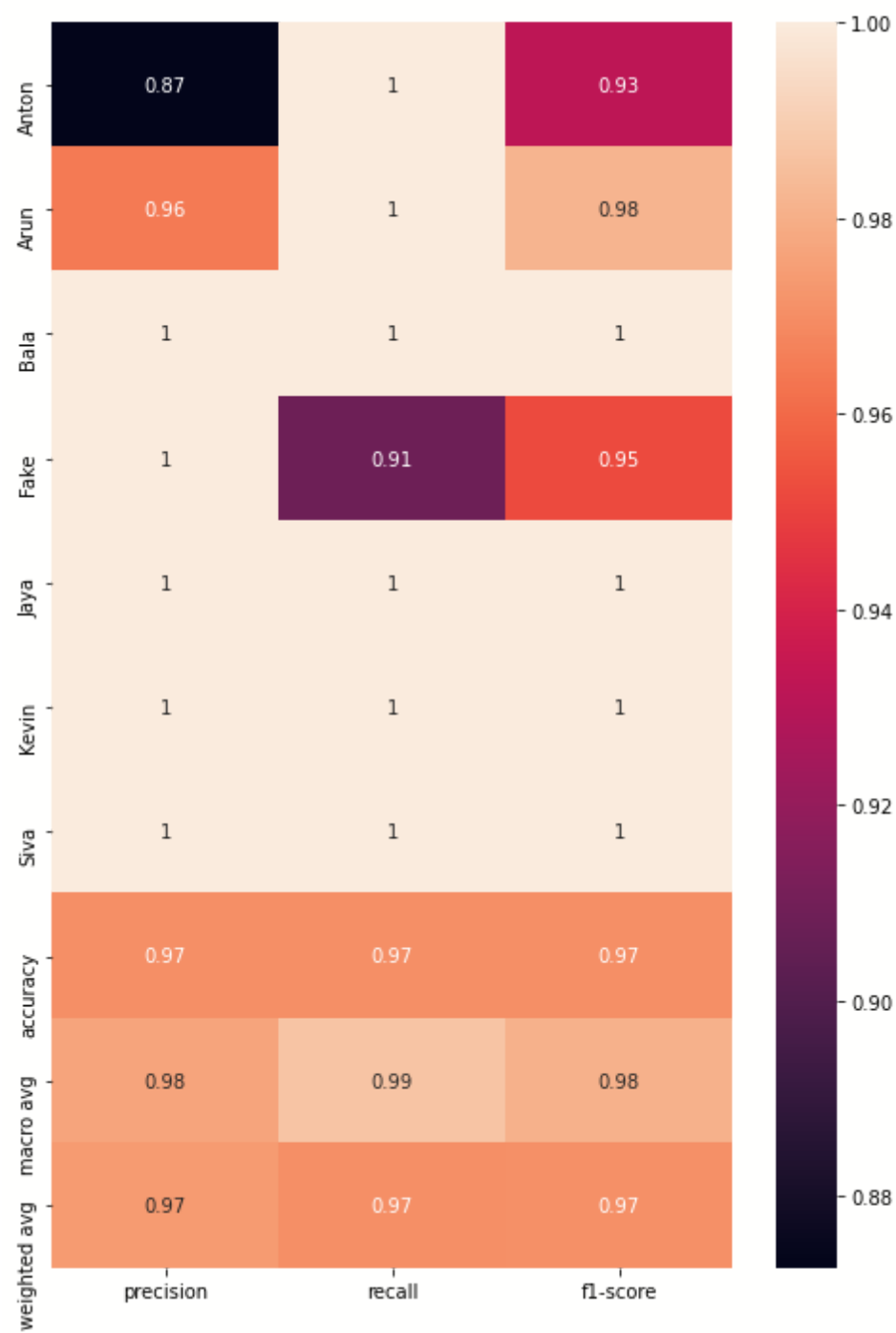


<p>Anton</p> 	<p>Anton</p> 
<p>Kevin</p> 	<p>Kevin</p> 
<p>Fake</p> 	<p>Fake</p> 
<p>Fake</p> 	<p>Fake</p> 

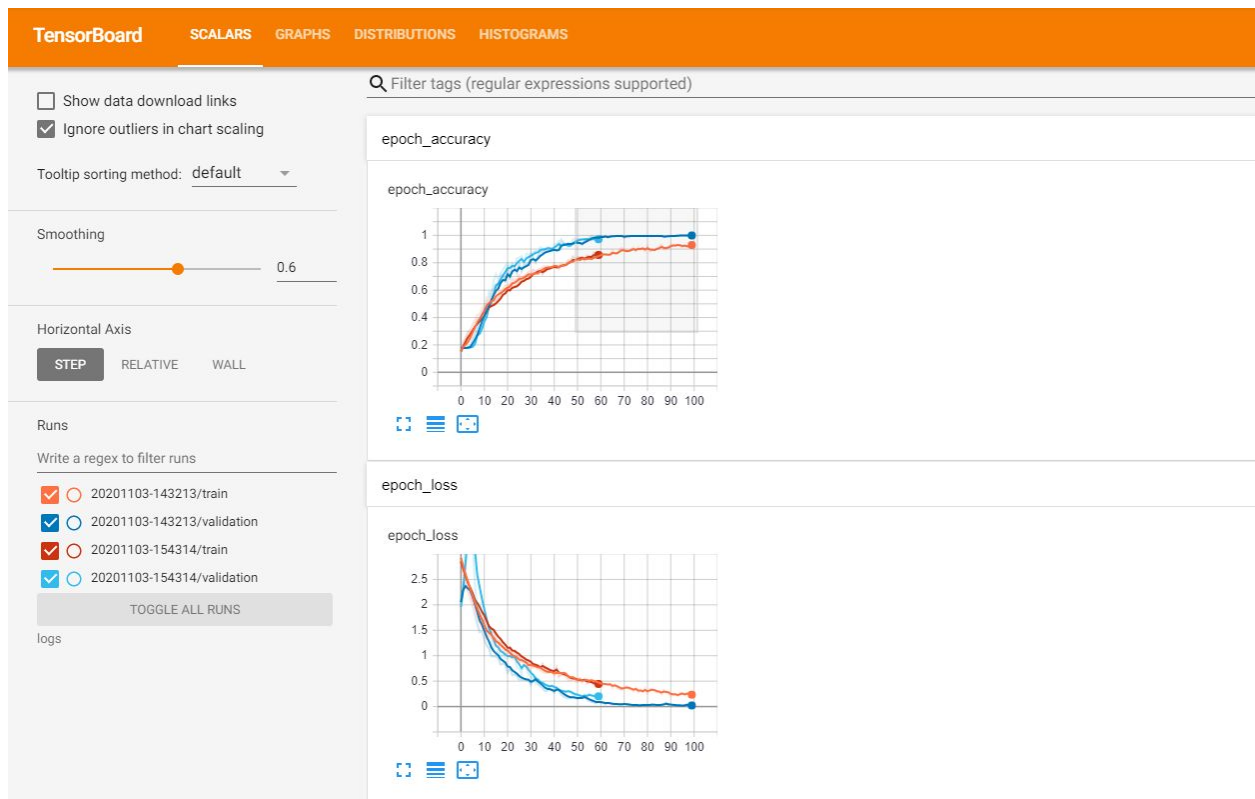
The following plot shows the number of extracted face ROIs for each class(real person, fake).



Classification Report



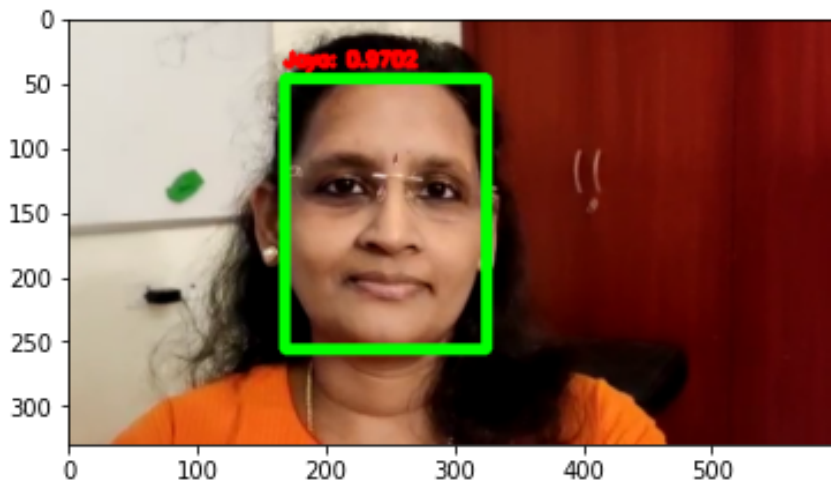
TensorBoard Logs



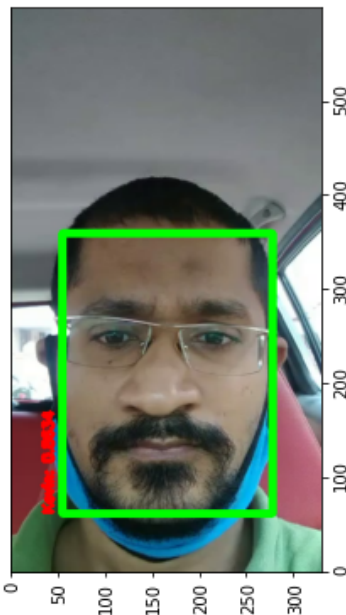
Result - Face Recognition

Result of the Face Recognition model is the identified person's name from the live video, with a bounding box over face and with confidence level mentioned.

Jaya 0.97020155



Kevin 0.8633561



Implications

The proposed Face Recognition model is able to recognize the known faces and fake images with greater accuracy, in real time using a web camera.

The advantage of using this model is, it can recognize faces with less training data. As it uses a shallow network with few parameters, the model can be run on a very small device, such as Raspberry Pi.

This also enables a contactless biometric identification in very less time.

The solution can be modified, or extended to use in other areas like, masked face detection, touchless attendance, emotion recognition, human ethnicity identification, defects detection in manufacturing, surveillance in crowded and large gathering places for public safety, etc,...

Limitations

Considering that the application of Intrusion Detection in the real world is for safety concerns and prevention of threat in any form, not everybody in the real world comes with threat. Therefore, labelling everybody to be an intruder and then disproving the same is an unnecessary load on the computation. It consumes time and effort of the admin in proving not an intruder.

The model is trained with the images of the admins at first. Anybody else apart from the admins are classified to be intruders at first because of the absence of image data.

Consider the following the real-world situation:

A person who is classified as an intruder at first will not be classified as an intruder the second time by the model. But in the real world there are possibilities where the person classified as an intruder at first can be an object of threat the second time. Safety concerns take a huge hit during these times.

How to enhance the solution under both these circumstances?

A person who is a potential threat may carry hazardous objects with him, certainly not under all conditions. The model can therefore be trained on the images of hazardous objects and non-hazardous objects as well. If a person (First appearance or Second appearance) with any hazardous objects (On which the model is trained) with him, is possibly a threat and a safety concern. This solution can simplify and improve the efficiency of the Intruder Detection System.

Conclusion

Thus we implemented a home security system with Intruder alert, there are some limitations in this system, which is, limited dataset, continuation of model training whenever new dataset is being generated, and communication to the admin / user of this system with applications like Slack

We have implemented LivenessNet model to identify fake / real persons and we have used transfer model of Caffemodel to identify the face positions

Future implementations

Next version will have the communicational channel with Slack, where Admin / User can connect with system's Slack channel and whenever any intruder identification, any suspicious moments, system will send message to that channel with the image captured, this will help to understand the scenario much better and faster

Supporting multiple camera inputs and analyse all inputs simultaneously, and also differentiate all inputs, which will help to identify the location of the incident happening, like main door, balcony etc

Literature survey

There are many approaches implemented in Face Liveness Detection. In this section, some of the most interesting liveness detection methods are presented.

Frequency and Texture based analysis

This approach is used by Gahyun Kim et al [1]. The basic purpose is to differentiate between live face and fake face (2-D paper masks) in terms of shape and detailedness. The authors have proposed a single image-based fake face detection method based on frequency and texture analyses for differentiating live faces from 2-D paper masks. The authors have carried out a power spectrum based method for the frequency analysis, which exploits both the low frequency information and the information residing in the high frequency regions.

Similar technique of face spoofing detection from single images using micro-texture analysis was implemented by Jukka et al. [2]. The key idea is to emphasize the differences of micro texture in the feature space. The authors adopt the local binary patterns (LBP) which is a powerful texture operator, for describing the micro-textures and their spatial information. The vectors in the feature space are then given as an input to an SVM classifier which determines whether the micro-texture patterns characterize a fake image or a live person image.

Variable Focusing based analysis

The technique of face liveness detection using variable focusing was implemented by Sooyeon Kim et al. [3]. The key approach is to utilize the variation of pixel values by focusing between two images sequentially taken in different focuses which is one of the camera functions. Assuming that there is no big difference in movement, the authors have tried to find the difference in focus values between real and fake faces when two sequential images(in/out focus) are collected from each subject.

Movement of the eyes based analysis

The technique based on the analysis of movement of eyes was introduced by Hyung-Keun Jee et al. for embedded face recognition system [4]. The authors proposed a method for detecting eyes in sequential input images and then variation of each eye region is calculated and whether the input face is real or not is determined. The basic assumption is that because of blinking and uncontrolled movements of the pupils in human eyes, there should be big shape variations.

Optical Flow based analysis

The method based on optical flow field was introduced by Bao et al. [5]. It analyzes the differences and properties of optical flow generated from 3D objects and 2D planes. The motion of an optical flow field is a combination of four basic movement types: Translation, rotation, moving and swing. The authors found that the first three basic types are generating quite similar optical flow fields for both 2D and for 3D images. The fourth type creates the actual differences in optical flow fields. Their approach is basically based on the idea that the optical flow field for 2D objects can be represented as a projection transformation.

A combination of face parts detection and an estimation of optical flow field for face liveness detection were introduced by Kollreider et al. [6]. This approach is able to differentiate between motion of points and motion of lines. The authors have suggested a method which analyzes the trajectories of single parts of a live face. The information which is being obtained can be used to decide whether a printed image was used or not. This approach uses a model-based Gabor decomposition and SVM for detection of face parts. The basic idea of this method is based on the assumption that a 3D face generates a 2D motion which is higher at central face regions than at the outer face regions such as ears. Therefore, parts which are farther away move differently from parts which are nearer to the camera. But, a photograph generates a constant motion on different face regions. With the information of the face parts positions and their velocity, it is possible to compare how fast they are in relation to each other [17]. This information is used to differentiate between a live face from a photograph.

Blinking based analysis

The blinking-based approach for liveness detection using Conditional Random Fields (CRFs) was introduced by Lin Sun et al.[7]. The authors have used CRFs to model blinking activities, for accommodating long-range dependencies on the observation sequence. Then they compared the CRF model with a discriminative model like AdaBoost and a generative model like HMM. Conditional random fields(CRFs) are probabilistic models for segmenting and labeling sequence data and mainly used in natural language processing for its accommodating long-range dependencies on the observation sequence. Blinking activity is an action represented by the image sequence which consists of images with close and non-close state

References

- [1] G. Kim, S.Eum, J. K. Suhr, D. I. Kim, K. R. Park, and J. Kim, Face liveness detection based on texture and frequency analyses, 5th IAPR International Conference on Biometrics (ICB), New Delhi, India. pp. 67-72, March 2012.
- [2] J. Maatta, A. Hadid, M. Pietikainen, Face Spoofing Detection From Single images Using MicroTexture Analysis, Proc. International Joint Conference on Biometrics (UCB 2011), Washington, D.C., USA.
- [3] sooyeon Kim, Sunjin Yu, Kwangtaek Kim, Yuseok Ban, Sangyoun Lee, Face liveness detection using variable focusing, Biometrics (ICB), 2013 International Conference on, On page(s): 1 – 6, 2013.
- [4] H. K. Jee, S. U. Jung, and J. H. Yoo, Liveness detection for embedded face recognition system, International Journal of Biological and Medical Sciences, vol. 1(4), pp. 235-238, 2006.
- [5] Wei Bao, Hong Li, Nan Li, and Wei Jiang, A liveness detection method for face recognition based on optical flow field, In Image Analysis and Signal Processing, 2009, IASP 2009, International Conference on, pages 233 –236, April 2009.
- [6] K. Kollreider H. Fronthaler, and J. Bigun, Evaluating liveness by face images and the structure tensor, in Proc of 4th IEEE Workshop on Automatic Identification Advanced Technologies, Washington DC, USA, pp. 75-80, October 2005.
- [7] Lin Sun, Gang Pan, Zhaohui Wu, Shihong Lao, Blinking-Based Live Face Detection Using Conditional Random Fields, ICB 2007, Seoul, Korea, International Conference, on pages 252-260, August 27-29, 2007.
- [8] Gang Pan, Zhaohui Wu and Lin Sun, Liveness Detection for Face Recognition, Recent Advances in Face Recognition, I-Tech, on Page(s): 236, December, 2008.
- [9] Jianwei Yang, Zhen Lei, Shengcai Liao, Li, S.Z, Face Liveness Detection with Component Dependent Descriptor, Biometrics (ICB), 2013 International Conference on Page(s): 1 – 6, 2013.
- [10] Andrea Lagorio, Massimo Tistarelli, Marinella Cadoni, Liveness Detection based on 3D Face Shape Analysis, Biometrics and Forensics (IWBF), 2013 International Workshop on Page(s): 1-4, 2013.