Celebrity Video Face Recognition

Ngoc-Vien DANG
EURECOM
Ngoc-Vien.Dang@eurecom.fr

Thanh-Long NGUYEN
EURECOM
Thanh-Long.Nguyen@eurecom.fr

Lucas PASCAL
EURECOM
Lucas.Pascal@eurecom.fr

Alison REBOUD
EURECOM
Alison.Reboud@eurecom.fr

Benoit HUET
EURECOM
Benoit.Huet@eurecom.fr

ABSTRACT

This project proposes the approaches for human face recognition, which focus on celebrities or public figures such as politicians, singers, or actors. The system starts training with an initial small data-set of celebrities' photos with the pre-train MTCNN model for face detection. Next, the outcome of the previous stage will be used for face recognition task via pre-train FaceNet model. The pre-train FaceNet model was trained under Inception-ResNet-v1 Architecture with VGGFace2 data-set. In order to test the face recognition accuracy, we have experimented with the embedding output from FaceNet model with different classifiers such as Random Forest, KNN, SVC or LinearSVC. Later, we also introduced face recognition for new celebrities' faces, which do not include in the data-set and using S.O.R.T algorithm to improve the accuracy. This training strategy has achieved a solid performance in face recognition in testing videos

1 INTRODUCTION

In recent years, Face recognition systems have strong growth and have been explored in several ways, such as human identification, video surveillance, celebrity spotting, etc. Furthermore, the continuous new studies and expansions of deep-learning based approaches have also supported this trend as the accuracy outcome of various face recognition problems have been improved significantly.

This project focuses on human face recognition, especially celebrities or public figures such as are politicians, singers, or actors by using a small set of labeled training data. Then an automatic system will be built to recognize the celebrities' faces in a video. Furthermore, the system have to be able to detect the new faces that have not existed in the training data-set.

This paper describes our training approaches and experiment's results for the tasks mentioned above. The paper's main structure can be considered as follow: Section 2 detailed about algorithms, models and libraries, which have been used in the paper as Multi-task Cascaded Convolutional Network in 2.2 or Inception-ResNet-v1 Architecture in 2.3.3. Next, the section 3 will explain about the general training strategies of this project and the further improvement for new the face recognition, which are not included the training data-set as in section 4 and the accuracy enhancement by using Simple Online and Realtime Tracking - S.O.R.T Algorithm in section 5. We describe the experimental results in section 6. Finally, in Section 7 we provide the conclusions.

2 BACKGROUND

2.1 Tools and Libraries

The major libaries and architectures we have used are:

- Tensorflow
- Scikit-learn
- OpenCV
- Multi-task Cascaded Convolutional Network (MTCNN)
- Inception-ResNet-v1 Architecture
- FaceNet
- Simple Online and Realtime Tracking (SORT)

2.2 Multi-task Cascaded Convolutional Network (MTCNN)

Multi-task Cascaded Convolutional Neural Networks (MTCNN) is an algorithm consisting of 3 stages, which detects the bounding boxes of faces in an image along with their 5 Point Face Landmarks [11]. Each stage gradually improves the detection results by passing its inputs through a CNN, which returns candidate bounding boxes with their scores, followed by non max suppression. Those three stages summary has depicted in the figure 1 and specific in the next 2

- Stage 1: A fully convolutional network which is called Proposal Network (P-Net) is used to obtain proposed regions and their bounding box regression vectors. The obtained regression vectors are used to calibrate the proposed regions and then apply non-maxima suppression (NMS) to merge highly overlapped regions.
- Stage 2: All proposed regions will be fed to another CNN which is called Refine Network (R-Net), which will reject a large number of false candidates, performs another calibration with bounding box regression and also NMS candidate merge.
- Stage 3: In the last stage, it is similar to the second stage and is called Output Network (O-Net). To furthermore describe face in details, they also output five facial landmarks positions.

2.3 Inception-ResNet-v1 Architecture

2.3.1 Inception Network. Inception Network is a deep convolutional neural network architecture codenamed Inception [9]. Inception Network was once considered a state-of-the-art deep learning architecture (or model) for solving image recognition and detection problems. The paper proposes a new idea of creating deep architectures. This approach helps to maintain the "computational

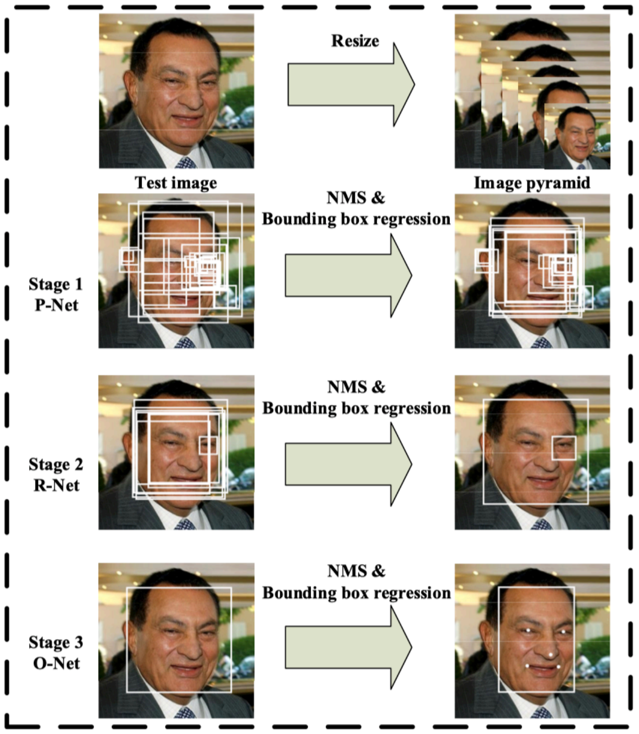

Figure 1: Three stages of MTCNN Structure
from [11]

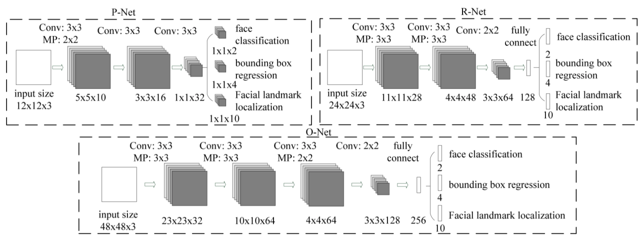

Figure 2: Overall Architecture of P-Net, R-Net, and O-Net

budget", while increasing the depth and width of the network. It is basically a convolutional neural network (CNN) which is 27 layers deep and it includes inception layers, which is a combination of all those layers (namely, 1×1 Convolutional layer, 3×3 Convolutional layer, 5×5 Convolutional layer) with their output filter banks concatenated into a single output vector forming the input of the next stage. Furthermore, the inception layer also has extra 1×1 Convolutional layer before applying another layer, which is mainly used for dimensionality reduction and a parallel Max Pooling layer, which provides another option to the inception layer.

In short, a deep learning model will have a layer that has learned to focus on individual parts of a face. The next layer of the network would probably focus on the overall face in the image to identify the different objects present there. In order to achieve it, the layer should have the appropriate filter sizes to detect different objects, and it is what inception layer should do. It allows the internal layers to pick and choose which filter size will be relevant to learn the required information. So even if the size of the face in the image is different, the layer works accordingly to recognize the face.

2.3.2 ResNet. This network architecture is based on the deep residual framework, which uses shortcut connections [3]. ResNet-X means Residual deep neural network with X number of layers, for instance: ResNet-101 means Resnet constructed using 101 layers. The major problem that Resnet solved is the vanishing gradient, with ResNets, the gradients can flow directly through the skip connections backwards from later layers to initial filters. Next, the core idea of ResNet is introducing a so-called "identity shortcut connection" that skips one or more layers, as while which stacking layers should not degrade the network performance, because we could simply stack identity mappings (layer that does not do anything) upon the current network, and the resulting architecture would perform the same. It indicates that the deeper model should not produce a training error higher than its shallower counterparts. They hypothesize that letting the stacked layers fit a residual mapping is easier than letting them directly fit the desired underlying mapping. And the residual block above explicitly allows it to do precisely that.

The ResNet version that deployed in the Inception ResNet v1 Architecture is the ResNet v3 [10], which was proposed a number of upgrades which increased the accuracy and reduced the computational complexity.

2.3.3 Inception-ResNet-v1. Inspired by the performance of the ResNet, a hybrid inception module was proposed in the paper [8] as It took the best of both worlds in terms of architectural design and added the ResNet aspect to the Inception architecture. One of them is Inception ResNet v1, which has a computational cost that is similar to that of Inception v3 and the introduction of three main inception modules, named A, B and C (Figure 3). The major idea as it introduced residual connections that add the output of the convolution operation of the inception module, to the input. The final network layout for Inception-ResNet-v1 is in figure 4

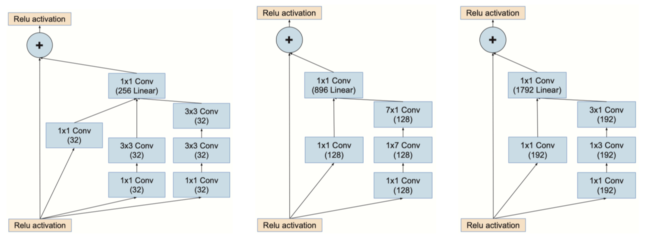

Figure 3: The schema for interior grid modules of the Inception-ResNet-v1 network

2.4 Training Data-sets

2.4.1 VGGFace2 Data-set. The VGGFace2 has described a follow-up work of their 2017 paper titled "VGGFace2: A data-set for recognizing faces throughout pose and age" [2]. The data-set incorporates 3.31 million pictures of 9, 131 topics, with a mean of 362.6 pictures for every topic. Photos are downloaded from Google Picture Search

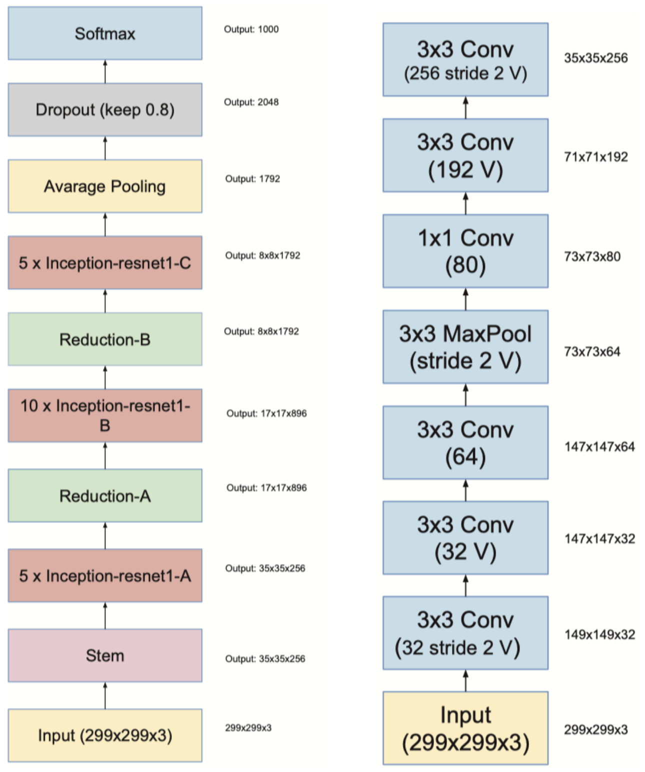

Figure 4: The overall schema for the Inception-Resnet-v1

and have massive variations in pose, age, illumination, ethnicity and occupation such as actors, athletes, politicians.

2.4.2 LFW Data-set. The LFW data-set was introduced in the paper named "Labeled Faces in the Wild: A Database for Studying-Face Recognition in Unconstrained Environments" [4]. The data-set included 13, 233 target face images from 5, 749 different individuals, whenever, 1, 680 people have two or more images in the database and the remaining 4, 069 people have just a single image in the database. The images are available as 250 by 250 pixel JPEG images. The database exhibits "natural" variability in the pose, lighting, focus, resolution, facial expression, age, gender, race, accessories, make-up, occlusions, background, and photographic quality. Despite this variability, the images in the database are presented in a simple and consistent format for maximum ease of use.

2.5 The Pre-train FaceNet Model with VGGFace2 data-set

2.5.1 FaceNet Model Basic Structure. FaceNet [7] is a deep learning model, which used the Inception-Resnet-v1 architecture. It directly learns a mapping from face images into a compact Euclidean space where distances directly correspond to a measure of face similarity. Once these embeddings are created then procedures like face recognition and verification can be done utilizing these embeddings as features.

FaceNet uses convolutional layers to learn representations directly from the pixels of the face. This network was trained on a large dataset to attain invariance to illumination, pose, and other variable conditions. FaceNet creates a 128-dimensional embedding from images and inserts them into a feature space, in such a way, that the squared distance between all faces, regardless of the imaging conditions, of the same identity, is small, whereas the squared distance between a pair of face images from distinct characters is large. The figure 5 depicts the model architecture.

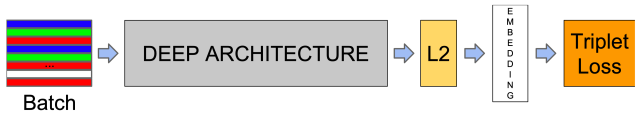

Figure 5: The FaceNet Model basic structure

The whole Deep Learning Architecture includes:

- 22 layers:
  - 11 convolutions
  - 3 normalizations
  - 4 max-pooling
  - 1 concatenation
  - 3 fully-connected
- 140 million parameters.

2.5.2 Embedding. Transform an image to a low dimensional feature space ($128d$) as it is known for Natural Language Processing (NLP). An embedding is a collective name for mapping input features to vectors. In a facial recognition system, these inputs are images containing a subject's face, mapped to a numerical vector representation.

Since these vector embeddings are represented in shared vector space, vector distance can be used to calculate the similarity between two vectors. In a facial recognition context, this can vector distance be applied to calculate how similar two faces are. Additionally, these embeddings can be used as feature inputs into a classification, clustering, or regression task.

2.5.3 Loss Function: Softmax Loss. This system employs a particular loss function called Softmax. Softmax function takes an N-dimensional vector of real numbers and transforms it into a vector of real numbers in the range (0,1) which add up to 1.

$$p_i = \frac{\exp^{a_i}}{\sum_{k=1}^{N} \exp_k^a}$$

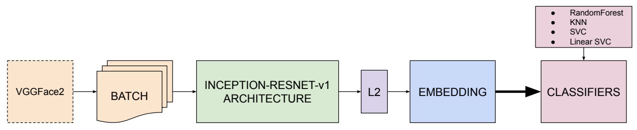

Figure 6: The Structure of FaceNet model with VGGFace2 data-set and different Classifiers

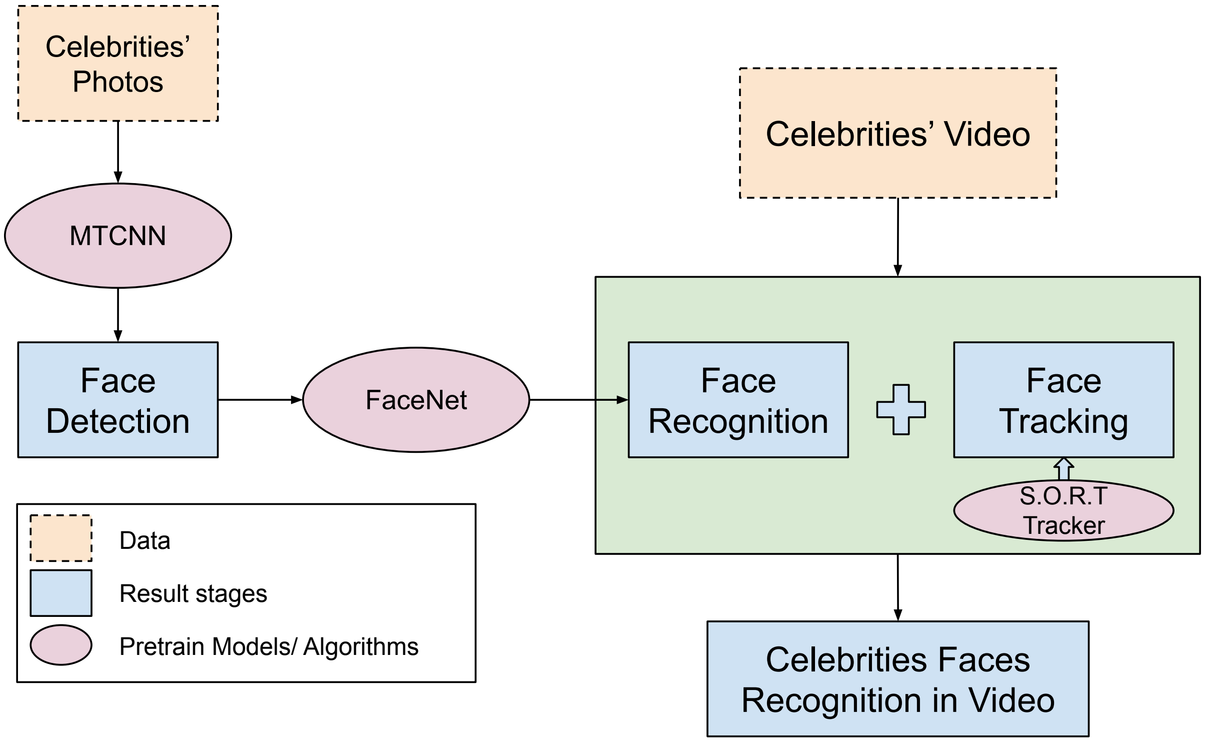

Figure 7: The Project Proposed Approach

2.5.4 The Pre-train FaceNet Model with VGGFace2 data-set. We can use FaceNet to create embeddings for faces of our own choice and then train a classifier (Random Forest, KNN, SVC or Linear SVC) to use these embeddings and do the classification. The proposed method of this project use Pre-train FaceNet model with VGGFace2 data-set, which has been trained and proposed by David Sandberg [6] to create embeddings for faces recognition and then train with various classifiers (Random Forest, KNN, SVC and Linear SVC) to use these embeddings and do the classification. The figure 6 has illustrated our approach.

2.6 Simple Online and Realtime Tracking - S.O.R.T Algorithm

Simple Online and Realtime Tracking (SORT) is an object tracking algorithm, which 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 [1].

The idea is to use some off-the-shelf model for object detection and then plug the results into the SORT algorithm that matches detected objects across frames. This approach obviously yields a multi-purpose algorithm: SORT does not need to know which type of object we track. It does not even need to learn anything: to perform the associations SORT uses mathematical heuristics such as maximizing the IOU (intersection-over-union) - the distance between each detection and all predicted bounding boxes from the 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. The quality of such an approach naturally depends a lot on the quality of the underlying object detection.

The implementation of S.O.R.T algorithm on this paper was inspired by the suggestion code from Linzaer [5] whenever it used the face detection results from MTCNN for the initial input for the face-tracking process. The

3 PROPOSED APPROACH

We propose our training approach, which can be refer to the graph in figure 7, with following stages:

1. **Face Detection**: We use the Pre-train MTCNN, which have been trained and provided here [11] and using the LFW data-set as the training input.
2. **Face Recognition**: Then we will plug the previous stage results into the Pre-train FaceNet Model, which has been described in the section 2.5. We have experimented with various classifications such as Random Forest, KNN, SVC and Linear SVC. The results of the experiments were presented in section 6.1 and section 6.2
3. **Face Tracking**: The outputs of Face Recognition stage were used together with the SORT Tracker algorithm in order to improve the face recognition performance within the video
4. **Final outcome**: a video will be treated as input of the combination models with face recognition of FaceNet model and S.O.R.T for the final face recognition output. It result can be referred to the section 6.3

4 AUTO-TRAINING FOR NEW PERSONS

In practice, the appearance of new persons happens quite often. We apply this approach to automatically training classifiers for persons not in the database. The embedding of new persons is merged with the existing embedding by merging and mapping 2 files. And we train the classifier for the new embedding file. The random forest (RF)and Support Vector Machine (SVM) classifier are applied. We also have tested the convergence speed and evaluated the performance of these new classifiers. These experimental results are shown in the Experimental Results part.

5 PERFORMANCE IMPROVEMENT: TRACKER AND FACENET

We have used the Simple Online and Real-Time tracking (SORT) algorithm to track every face. The FaceNet is also applied to every detected frame. We tracked the face from the beginning it detected and assigned it to the object ID until the tracker lost that ID, and used Facenet to find out the label (the class name) for that ID in all frames having that ID. The Object ID and the temporary label for every face will be generated. After that, the system will try to guess the label for each face by using the majority rule. The output file containing the bounding box, label name, and the frame order for every face will be generated and stored and the predicted labels also will be printed on the frame.

6 EXPERIMENTAL RESULTS

We evaluate our system on the commonly used and large dataset, Labeled Faces in the Wild (LFW). Therefore the metric we use to deal with classification problem is the f1-score.

First, we are interested in comparing the performance of the K-Nearest Neighbor (KNN), Support Vector Machine (SVM), and Random Forest classifier when only one or few training samples are provided. Second, we are interested in the convergence rate of these classifiers when adding a new person into an existing embedding. Finally, we are interested in the performance of the system when combination FaceNet with tracker with video as input.

6.1 Test Case 01:

Comparing the performance of the K-Nearest Neighbor (KNN), Support Vector Machine (SVM), and Random Forest classifier when only one or few training samples are provided.

Based on the evaluation results from the tables above, we have the following comments:

- KNN worked really well even when only have a single image.
- The accuracy of SVM when only a few samples were provided is really low (even reached 0.000 % when only one provided) like as the paper "Incremental Training for Face Recognition" mentioned.
- And the accuracy of Random Forest even reached about 34.40% (in LFW dataset) with a single image and gradually increased when more samples are provided.

Min. No. image per class	No. training image per class	SVM	KNN	RF
40	35	0.999	1	0.998
5	**1**	**0**	**0.92**	**0.344**
10	1	0	0.952	0.771
20	1	0	0.982	0.927
5	**2**	**0.282**	**0.972**	**0.44**
10	2	0.348	0.98	0.82
20	2	0.351	0.991	0.974
5	**3**	**0.942**	**0.977**	**0.543**
10	3	0.908	0.985	0.857
20	3	0.99	0.993	0.986

Table 1: Evaluation results on the LFW data-set

6.2 Test Case 2:

Comparing the convergence rate of the SVM and Random Forest classifiers when adding a new person into an existing embedding. The new person used for testing is **Alejandro Toledo**:

Alejandro_Toledo _0012.jpg
Alejandro_Toledo _0013.jpg
Alejandro_Toledo _0014.jpg
Alejandro_Toledo _0015.jpg
Alejandro_Toledo _0016.jpg

Figure 8: Alejandro Toledo

Creating the scenarios for testing as following:

- **Scenario 01:** The existing embedding1 having:
  - Number of classes: 61
  - Number of images: 2,374
- **Scenario 2:** The existing embedding2 having:
  - Number of classes: 142
  - Number of images: 3,425
- **Scenario 3:** The existing embedding3 having:
  - Number of classes: 900
  - Number of images: 6,667

6.2.1 *Scenario 01:*

The result on the scenario with the existing embedding1 having 61 classes and 2,374 images. Each class contains at least 19 images.

SVC (kernel="linear")	Before	After but just 61-old persons	After with 62 persons
F-measure	0.995069	0.995069	0.995149
Training time (s)		7.569466352	

Table 2: The result of scenario 01 with SVM Classifier

6.2.2 *Scenario 02:*

The result on the scenario with the existing embedding2 having 142 classes and 3,425 images. Each class contains at least 5 images.

Random Forest	Before	After but just 61-old persons	After with 62 persons
F-measure	0.988395	0.988471	0.988657
Training time (s)		10.6547689437866	

Table 3: The result of scenario 01 with Random Forest Classifier

SVC (kernel="linear")	Before	After but just 61-old persons	After with 62 persons
F-measure	0.936096	0.936096	0.936542
Training time (s)		23.1906087398529	

Table 4: The result of scenario 02 with SVM Classifier

Random Forest	Before	After but just 61-old persons	After with 62 persons
F-measure	0.963136	0.960255	0.960533
Training time (s)		136.997495889663	

Table 5: The result of scenario 02 with Random Forest Classifier

6.2.3 Scenario 03: The result on the scenario with the existing embedding3 having 900 classes and 6,667 images. Each class contains at least 1 image.

SVC (kernel="linear")	Before	After but just 61-old persons	After with 62 persons
F-measure	0.337313	0.333806	0.333879
Training time (s)		205.333515167236	

Table 6: The result of scenario 03 with SVM Classifier

For the Random Forest classifier in this case: The dataset having the least populated classes has one member, which is too few. Therefore, RandomizedSearchCV from Sklearn to optimize our hyperparameters made errors during fit. So we cannot find the optimum parameters for this classifier. And if we choose parameters randomly, the result is quite bad. Based on the evaluation results from the tables above, we have the following comments:

- The classification-time of KNN was fastest, followed by SVM and slowest was Random Forest.
- Furthermore, as we mentioned above, classification-time per sample of KNN was really fast because Facenet mapped from image space to embedding space previously. Therefore, update the data and train a new version of the system from scratch does not take time.

6.3 Test Case 03:

The performance of the system when combination FaceNet with tracker with video as input. We used a video for testing. There are 4936 frames after we speech up x3, it's really hard for us to get ground truth to compare with our system. Therefore we have not had the specific number f1-score to evaluate the performance of this approach. But it works well in almost frames.

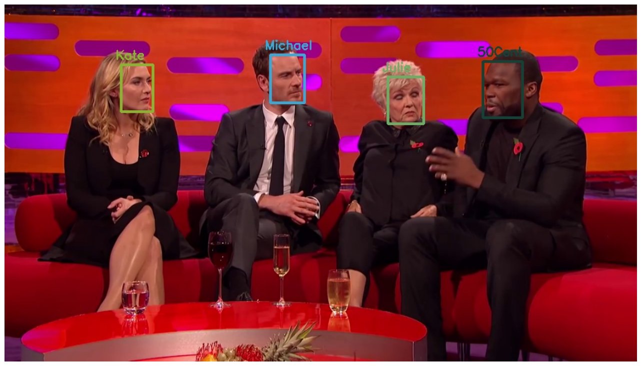

Figure 9: Sample Result

7 CONCLUSION AND FUTURE WORKS

As we can see that the proposed method of combination FaceNet and Tracker results in a good performance. Furthermore, we can use the SORT tracker to create the face training set from any videos.

In the future, we can perform extended research on finding the method to evaluate the performance for video and try incremental training approach such as online random forest as we were planning at the beginning.

REFERENCES

[1] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. 2016. Simple online and realtime tracking. In *2016 IEEE International Conference on Image Processing (ICIP)*. IEEE, 3464–3468.

[2] Qiong Cao, Li Shen, Weidi Xie, Omkar M Parkhi, and Andrew Zisserman. 2018. Vggface2: A dataset for recognising faces across pose and age. In *2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018)*. IEEE, 67–74.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[4] Gary B. Huang Erik Learned-Miller. 2014. *Labeled Faces in the Wild: Updates and New Reporting Procedures.* Technical Report UM-CS-2014-003. University of Massachusetts, Amherst.

[5] Linzaer. 2018. Face Detection & Tracking & Extract. (2018). https://github.com/Linzaer/Face-Track-Detect-Extract

[6] David Sandberg. 2018. Face Recognition using Tensorflow. (2018). https://github.com/davidsandberg/facenet

[7] Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 815–823.

[8] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. 2017. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-First AAAI Conference on Artificial Intelligence*.

[9] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1–9.

[10] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2818–2826.

[11] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao. 2016. Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks. *IEEE Signal Processing Letters* 23, 10 (Oct 2016), 1499–1503. DOI: https://doi.org/10.1109/LSP.2016.2603342