Face Detection, Tracking and Verification for Digital Signage Applications

A Project Report

submitted by

RAUNAK KALANI

in partial fulfilment of requirements
for the award of the dual degree of

BACHELOR OF TECHNOLOGY AND MASTER OF TECHNOLOGY

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY MADRAS

MAY 2018

THESIS CERTIFICATE

This is to certify that the thesis titled **Face Detection, Tracking and Verification for Digital Signage Applications**, submitted by **Raunak Kalani**, to the Indian Institute of Technology, Madras, for the award of the dual degree of **Bachelor of Technology and Master of Technology**, is a bona fide record of the research work done by him under my supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Prof. Kaushik Mitra
Research Guide
Assistant Professor
Dept. of Electrical Engineering
IIT-Madras, 600 036

Place: Chennai

Date: 9th May 2018

ACKNOWLEDGEMENTS

I would like to begin by thanking my guide, Prof. Kaushik Mitra for all his help, support and patience throughout the course of my project work. His approach and dedication to research has been incomparable and has always inspired me to keep pushing myself. All the students in the Computational Imaging Lab have been immensely cooperative, friendly and accommodating and have greatly inspired my own work. I would like to thank my project mate Akhil for helping me out through the project.
Were it not for the infallible faith of my parents and brother, I would not be present here. They have been a constant support throughout my life and I am forever grateful to them for their blessing and encouragement.
Lastly, I would like to give a huge shout-out to every one of my friends who have made my experience at IIT Madras, a treasured one.

ABSTRACT

KEYWORDS: Deep Learning ; Face Tracking; Face Detection; Re-Identification; Verification

The estimation of soft biometric features related to a person standing in front an advertising screen plays a key role in digital sign-age applications. Information such as gender, age, and emotions of the user can help to trigger dedicated advertising campaigns to the target user as well as it can be useful to measure the type of audience attending a store. To be able to do these tasks we need to first detect the faces of people standing in front of the screen. To get the features over the course of time of the advertisement we need to be able to track the person as he/she moves across the screen. In this project we describe a framework to be able to detect and track multiple faces in front of a camera. For the project to be deployable at multiple faces it needs to be cost effective and thus be able to work on medium cost machines.

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

CNN	Convolutional Neural Networks
SSH	Single Stage Headless (Face Detector)
YOLO	You Only Look Once
GOTURN	Generic Object Tracking Using Regression Networks
FTURN	Face Tracking Using Regression Networks
MOT	Multi Object Tracking
PoC	Proof of Concept

CHAPTER 1

Introduction and Motivation

Digital signage is considered a revolutionary research area which aims to build advanced technologies for the out-of-home advertising. More specifically, with the term "digital signage" are referred the smart screens employed to show advertising content to a broad audience in a public/private area (e.g., store, airport, info office, taxi, etc.). The advertising screens are usually connected to the Internet and are able to perform a series of "measurements" on the audience in front of the screen which are then exploited for marketing purposes (e.g., the screen reacts differently depending on the measurements). Thousands of organizations (e.g., retailers, government institutions, etc.) have already realized the benefits of the digital signage increasing the revenue related to their products or offering a better service in terms of given information to the audience.

In the context of digital signage, soft biometrics data inferred from the face of the user in front to an advertising screen (such as gender identification and age estimation) are used to collect information to be exploited for users profiling. Adhoc advertising campaigns are then showed, taking into account of the collected information. Recent works demonstrate that computer vision techniques for face detection, age and gender recognition, classification and recognition of people's behavior can provide objective measurements (e.g., time of attention) about the people in front of a smart display. Systems able to learn audience preferences for certain content can be exploited to compute the expected view time for a user, in order to organize a better schedule of the advertising content to be shown. The audience emotional reaction can be also captured and analyzed to automatically understand the feeling of the people to a campaign (e.g., to understand the attractiveness of a campaign with respect to another). Recent studies demonstrate that through computer vision methods it is possible quantify the percentage of the people who looked-at the display, the average attention time (differentiating by gender), the age groups who are more most responsive to the dynamic or static content.

Person re-identification plays a important role in this to check if we are continuing to track the same person or have we lost him/her either because of occlusion or tracking failure and subsequent re-detection.

Although the explosion of the field, in both academia and industry, it seems that measurements about the re-identification of a person in front a smart screen has been not taken into account in the context of digital signage. The information collected with a re-identification engine could be useful to answer the following question: is a specific person back to the advertising screen within a time slot? In computer vision literature different methods for person re-identification have been proposed. However, differently of the application contexts belonging to digital signage where in most of the cases only the person face is acquired to measure the information, the classic re-identification (e.g., in surveillance) is based on the exploitation of features extracted considering global appearance of an individual (e.g., clothing). Few works consider the re-identification based only on the person face.
In the following sections we will detail all the key "ingredients" useful to build a Digital Signage system. All of the networks have been implemented in caffe (Jia *et al.*, 2014).

CHAPTER 2

Background: Convolutional Neural Networks

2.1 What are CNNs?

2.1.1 Intuition

Suppose we are tracking the position of a space ship with a laser sensor at discrete time intervals. Now suppose that the measurements are noisy because of mis-calibration of the sensors. To obtain a less noisy estimate of the position of the space ship, we would like to average several measurements. Now more recent measurements are more important and hence we would like to take a weighted average. In practice we only sum over a small window.

$$s_t = \Sigma_{a=0}^{n} x_{t-a} w_{-a}$$

where s_t denotes the spaceship position at time t, x are the measurements made at time t and w are the weights that are assigned to those measurements. This $\mathbf{w}$ is called a filter or a kernel.

This was a 1-d convolution, we can do a 2-d convolution as well. In 2-d convolution there is a 2-d kernel or $\mathbf{w}$ matrix. It slides over the 2-d input matrix and produces the output as given by

$$S_{ij} = \Sigma_{\lfloor a=-\frac{m}{2}\rfloor}^{\lfloor\frac{m}{2}\rfloor} \Sigma_{\lfloor b=-\frac{n}{2}\rfloor}^{\lfloor\frac{n}{2}\rfloor} I_{i-a,j-b} K_{\frac{m}{2}+a,\frac{n}{2}+b}$$

as illustrated in figure 2.1.

But images are usually 3 dimensional, with the third dimension comprising of the 3 channels: R,G, and B. The visualization in figure 2.2 shows an example of 3d kernel applied to an image to get a feature map.

Convolutional Neural Networks (CS231n Convolutional Neural Networks for Visual Recognition) are very similar to ordinary Neural Networks, they are made up of

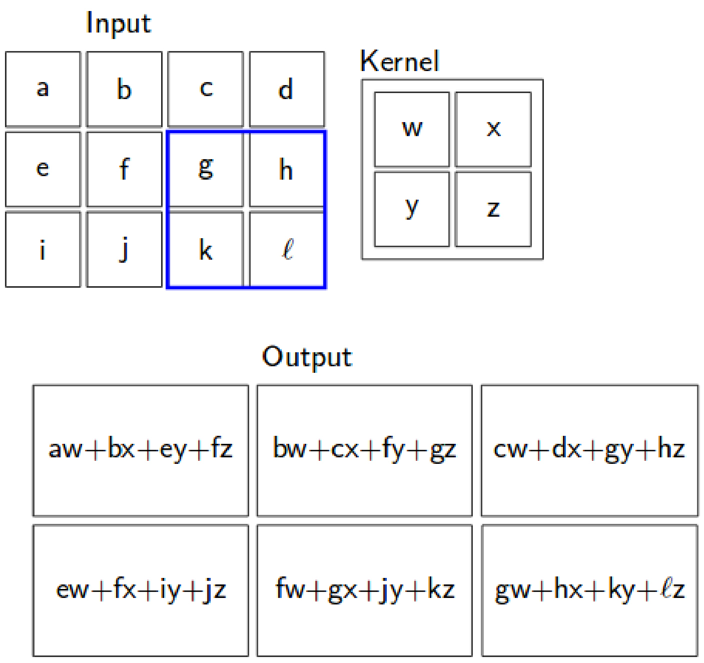


Figure 2.1: 2-d Convolution

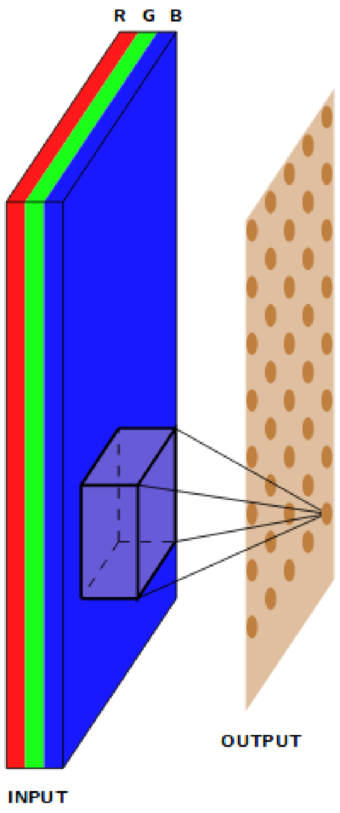


Figure 2.2: Convoltion on images

neurons that have learnable weights and biases. Each neuron receives some inputs, performs a dot product and optionally follows it with a non-linearity. The whole network still expresses a single differentiable score function: from the raw image pixels on one end to class scores at the other. And they still have a loss function (e.g. SVM/Softmax) on the last (fully-connected) layer and all the tips/tricks we developed for learning regular Neural Networks still apply. So what does change? ConvNet architectures make the explicit assumption that the inputs are images, which allows us to encode certain properties into the architecture. These then make the forward function more efficient to implement and vastly reduce the amount of parameters in the network.

2.1.2 Architecture Overview

Convolutional Neural Networks take advantage of the fact that the input consists of images and they constrain the architecture in a more sensible way. In particular, unlike a regular Neural Network, the layers of a ConvNet have neurons arranged in 3 dimensions: width, height, depth. (Note that the word depth here refers to the third dimension of an activation volume, not to the depth of a full Neural Network, which can refer to the total number of layers in a network.) For example, the input images in CIFAR-10 are an input volume of activations, and the volume has dimensions 32x32x3 (width, height, depth respectively). As we will soon see, the neurons in a layer will only be connected to a small region of the layer before it, instead of all of the neurons in a fully-connected manner. Moreover, the final output layer would for CIFAR-10 have dimensions 1x1x10, because by the end of the ConvNet architecture we will reduce the full image into a single vector of class scores, arranged along the depth dimension. Here is a visualization 2.3:

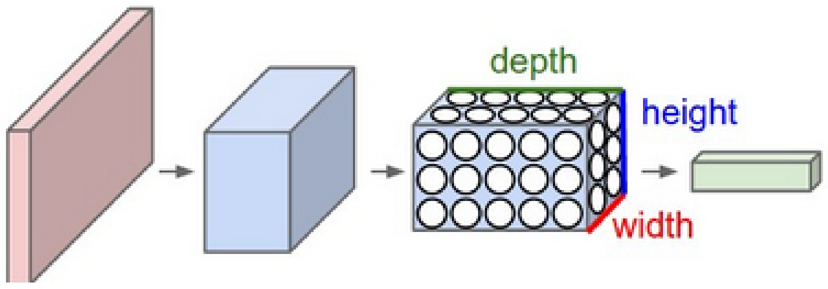


Figure 2.3: Convolution Network

2.1.3 Layers used in ConvNets

To explain the different layers of a typical ConvNet we will use the example of a network used to classify images into 10 categories as described in the CIFAR-10 dataset.

Input Layer

will hold the raw pixel values of the image, in this case an image of width 32, height 32, and with three color channels R,G,B.

Conv Layer

CONV layer will compute the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and a small region they are connected to in the input volume. This may result in volume such as [32x32x12] if we decided to use 12 filters.
The CONV layer's parameters consist of a set of learnable filters. Every filter is small spatially (along width and height), but extends through the full depth of the input volume. For example, a typical filter on a first layer of a ConvNet might have size 5x5x3 (i.e. 5 pixels width and height, and 3 because images have depth 3, the color channels). During the forward pass, we slide (more precisely, convolve) each filter across the width and height of the input volume and compute dot products between the entries of the filter and the input at any position. As we slide the filter over the width and height of the input volume we will produce a 2-dimensional activation map that gives the responses of that filter at every spatial position. Intuitively, the network will learn filters that activate when they see some type of visual feature such as an edge of some orientation or a blotch of some color on the first layer, or eventually entire honeycomb or wheel-like patterns on higher layers of the network. Now, we will have an entire set of filters in each CONV layer (e.g. 12 filters), and each of them will produce a separate 2-dimensional activation map. We will stack these activation maps along the depth dimension and produce the output volume.
When dealing with high-dimensional inputs such as images, as we saw above it is impractical to connect neurons to all neurons in the previous volume. Instead, we will connect each neuron to only a local region of the input volume. The spatial extent of

this connectivity is a hyperparameter called the receptive field of the neuron (equivalently this is the filter size). The extent of the connectivity along the depth axis is always equal to the depth of the input volume. It is important to emphasize again this asymmetry in how we treat the spatial dimensions (width and height) and the depth dimension: The connections are local in space (along width and height), but always full along the entire depth of the input volume.

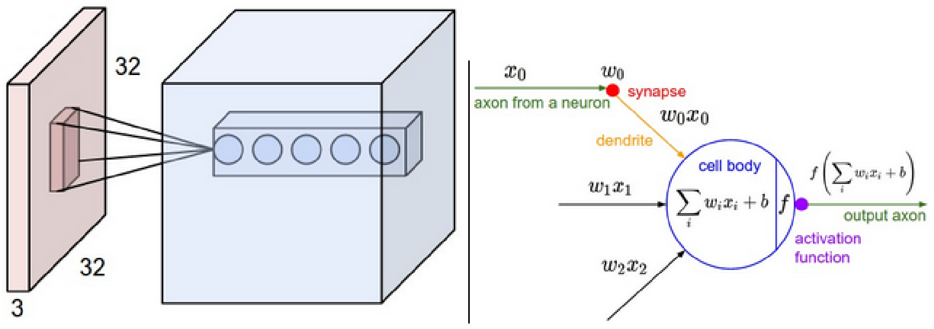


Figure 2.4: Conv layer

We need four hyper-parameters:

- Number of filters **K**
- Their spatial extent **F**
- Stride **S**
- The amount of zero padding **P**

to produce the output feature map of dimension:

- $\mathbf{W_2 = (W_1 - F + 2P)/S + 1}$
- $\mathbf{H_2 = (H_1 - F + 2P)/S + 1}$
- $\mathbf{D_2 = K}$

ReLu Layer

In the context of artificial neural networks, the rectifier is an activation function defined as the positive part of its argument:

$$f(x) = x^+ = max(0, x)$$

where x is the input to a neuron. This is also known as a ramp function and is analogous to half-wave rectification in electrical engineering. A unit employing the rectifier is also called a rectified linear unit (ReLU)

Pool Layer

It is common to periodically insert a Pooling layer in-between successive Conv layers in a ConvNet architecture. Its function is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network, and hence to also control overfitting. The Pooling Layer operates independently on every depth slice of the input and resizes it spatially, using the MAX operation. The most common form is a pooling layer with filters of size 2x2 applied with a stride of 2 downsamples every depth slice in the input by 2 along both width and height, discarding 75% of the activations. Every MAX operation would in this case be taking a max over 4 numbers (little 2x2 region in some depth slice). The depth dimension remains unchanged.

We need 2 hyper-parameters:

- Their spatial extent $\mathbf{F}$
- Stride $\mathbf{S}$

to produce the output feature map of dimension:

- $\mathbf{W_2 = (W_1 - F)/S + 1}$
- $\mathbf{H_2 = (H_1 - F)/S + 1}$
- $\mathbf{D_2 = D_1}$

Fully connected Layer

Neurons in a fully connected layer have full connections to all activations in the previous layer, as seen in regular Neural Networks. Their activations can hence be computed with a matrix multiplication followed by a bias offset.

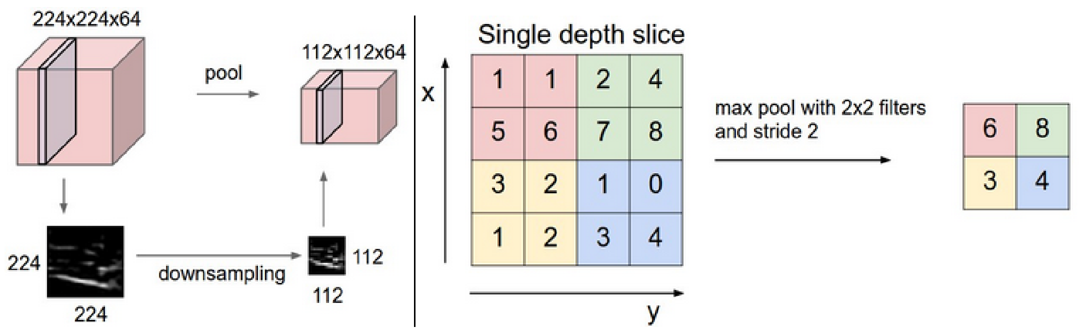
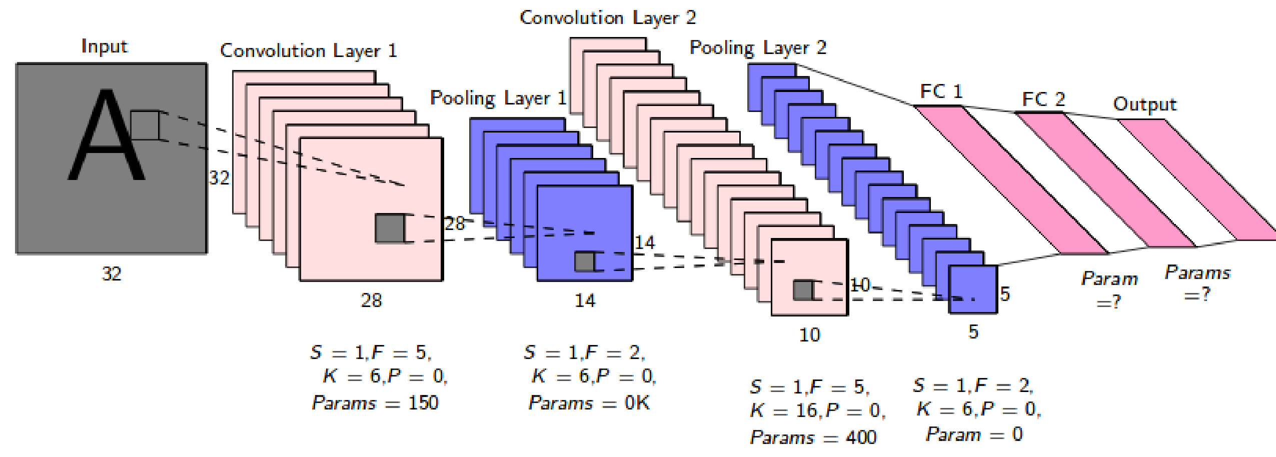


Figure 2.5: Pool Layer

In this way, ConvNets transform the original image layer by layer from the original pixel values to the final class scores. Note that some layers contain parameters and other don't. In particular, the CONV/FC layers perform transformations that are a function of not only the activations in the input volume, but also of the parameters (the weights and biases of the neurons). On the other hand, the RELU/POOL layers will implement a fixed function. The parameters in the CONV/FC layers will be trained with gradient descent so that the class scores that the ConvNet computes are consistent with the labels in the training set for each image. Figure 2.6 shows a typical ConvNet.



Input
A
32
32
Convolution Layer 1
28
28
$S = 1, F = 5, K = 6, P = 0, Params = 150$
Pooling Layer 1
14
14
$S = 1, F = 2, K = 6, P = 0, Params = 0K$
Convolution Layer 2
10
10
$S = 1, F = 5, K = 16, P = 0, Params = 400$
Pooling Layer 2
5
5
$S = 1, F = 2, K = 6, P = 0, Param = 0$
FC 1
FC 2
Output
$Param = ?$
$Params = ?$

Figure 2.6: Typical ConvNet

CHAPTER 3

Prior Work

3.1 Face Detection

To be able to get facial features of a person standing in front of a screen, we must be able to detect were the face is. For this we tested out multiple face detection algorithm namely the openCV face detection algorithm based on Haar Cascade method, YOLO (You Only Look Once), and SSH (Single Stage Headless) face detector. The last two are deep network based.

Below is a brief description of both of them.

3.1.1 YOLO9000: Better, Faster, Stronger

YOLO9000 (Redmon and Farhadi, 2016) gives a method for multiple object detection in real time in a video feed. The architecture is extremely fast. The base YOLO model processes images in real-time at 45 frames per second. A smaller version of the network, Fast YOLO, processes an astounding 155 frames per second while still achieving double the mAP of other real-time detectors. Compared to state-of-the-art detection systems, YOLO makes more localization errors but is less likely to predict false positives on background. Finally, YOLO learns very general representations of objects. It outperforms other detection methods, including DPM and R-CNN, when generalizing from natural images to other domains like artwork.The newer YOLO9000 can detect 9000 classes of object with more mAP than any other method while still working in real time. We trained the algorithm on face data from WIDER dataset to compare the algorithm with other face detection algorithm.

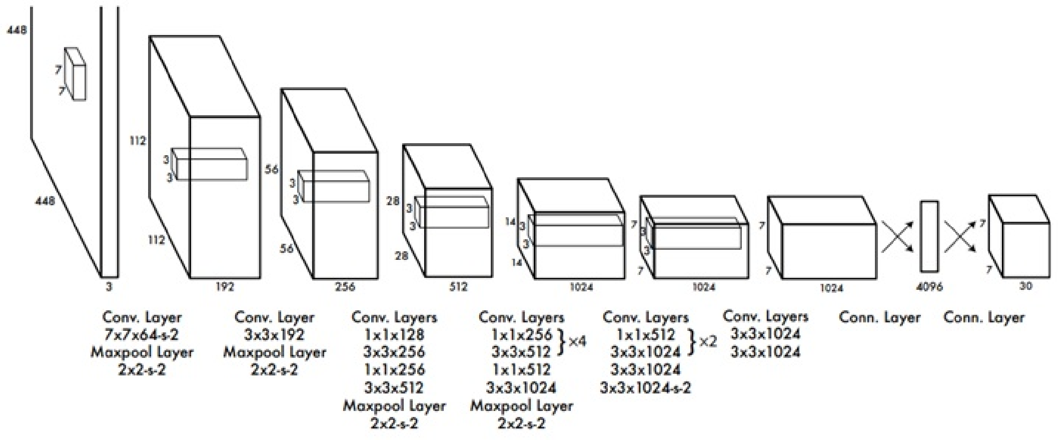


Figure 3.1: YOLO Architecture (Redmon and Farhadi, 2016)

3.1.2 SSH: Single Stage Headless Face Detector

SSH (Najibi *et al.*, 2017) introduces a new method for face detection. Unlike the conventional two-stage face detectors, SSH detects faces in a single stage directly from the early convolutional layers in a classification network. It is able to achieve state-of-the-art results while removing the "head" of its underlying classification network - i.e. all fully connected layers in the VGG-16 which contains a large number of parameters. The method is also able to detect multiple faces of different sizes simultaneously in a single pass. These properties make the network fast and light-weight. It is able to run at around 50 FPS.

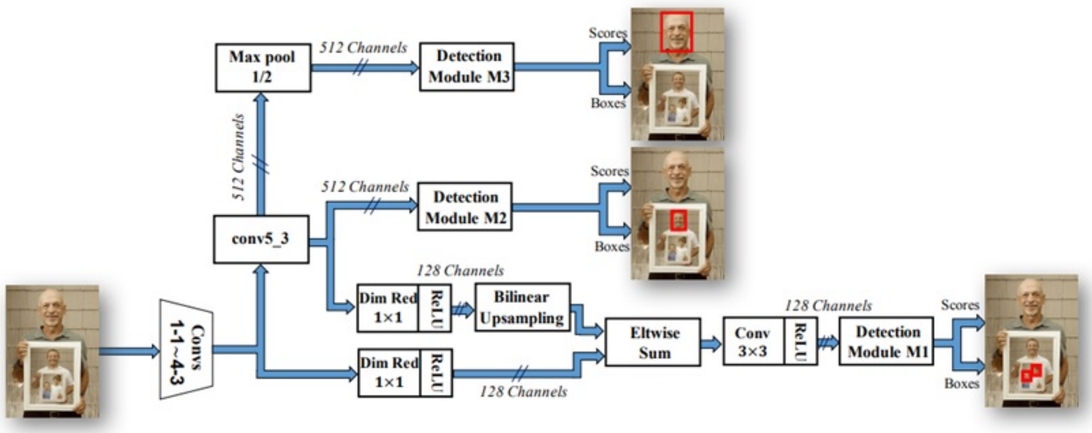


Figure 3.2: SSH Architecture (Najibi *et al.*, 2017)

To compare the above two algorithms we have used WIDER Dataset as a train and test

set.
Fig. 3.3 Shows some example detection by the two algorithms.

3.1.3 Datasets

The following two datasets were used for training and testing different algorithms proposed in this project.

WIDER Dataset

This dataset (Yang *et al.*, 2016) contains 32, 203 images with 393, 703 annotated faces, 158, 989 of which are in the train set, 39, 496 in the validation set and the rest are in the test set. The validation and test set are divided into "easy", "medium", and "hard" subsets cumulatively (i.e. the "hard" set contains all images). This is one of the most challenging public face datasets mainly due to the wide variety of face scales and occlusion. We train all models on the train set of the WIDER dataset and evaluate on the validation and test sets

VGG Face Dataset

The dataset (Parkhi *et al.*, 2015) consists of 2,622 identities. Each identity has an associated text file containing URLs for images and corresponding face detections. It is made available to the public by the University of Oxford. It contains images of celebrities in different poses and lightning conditions. This was used for training the tracking algorithm.

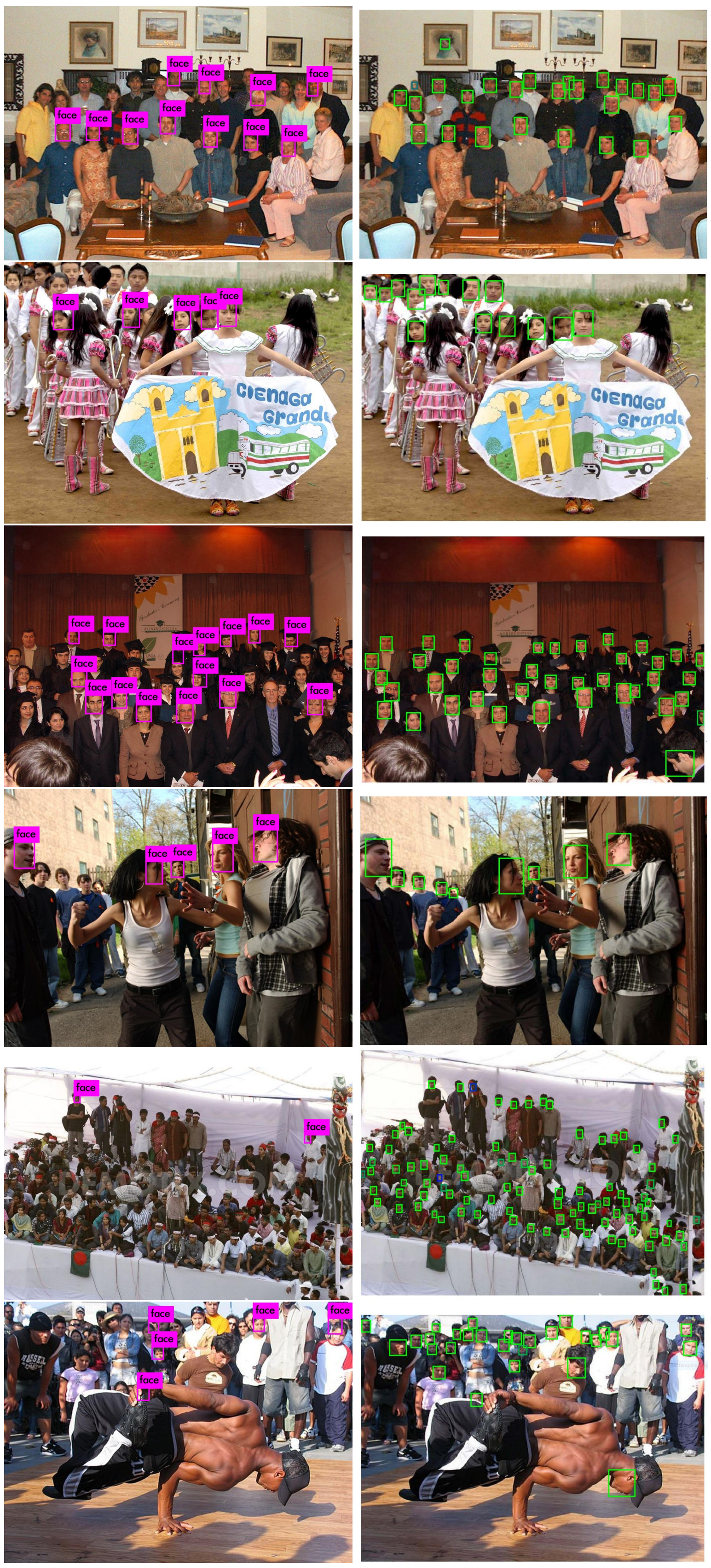


Figure 3.3: YOLO and SSH comparison on WIDER Dataset
Left: YOLO, Right: SSH

3.2 Face Tracking

The next goal is to track the detected face as the face can keep moving in front of the screen. To do this we need a robust face detector that can track faces as they are moving. We would like to use a tracker instead of trying to detect people in each frame because first, tracking is generally faster than detection. Secondly, if we detect in each frame we would have the problem of associating the features of the people in the current frame to that of the detections of the previous frame. This would further be computationally expensive and would further decrease the speed of tracking. Thus for tracking we start with a fast generic object tracker that can track single object in real time. The tracker is called GOTURN (Generic Object Tracking Using Regression Networks). Given some object of interest marked in one frame of a video, the goal of "singletarget tracking" is to locate this object in subsequent video frames, despite object motion, changes in viewpoint, lighting changes, or other variations. Single-target tracking is an important component of many systems.

3.2.1 GOTURN

GOTURN is General Object Tracking Using Regression Network (Held *et al.*, 2016). It is a siamese network. In this model, we input the target object as well as the search region each into a sequence of convolutional layers. The output of these convolutional layers is a set of features that capture a high-level representation of the image. The outputs of these convolutional layers are then fed through a number of fully connected layers. The role of the fully connected layers is to compare the features from the target object to the features in the current frame to find where the target object has moved. Between these frames, the object may have undergone a translation, rotation, lighting change, occlusion, or deformation. The function learned by the fully connected layers is thus a complex feature comparison which is learned through many examples to be robust to these various factors while outputting the relative motion of the tracked object.

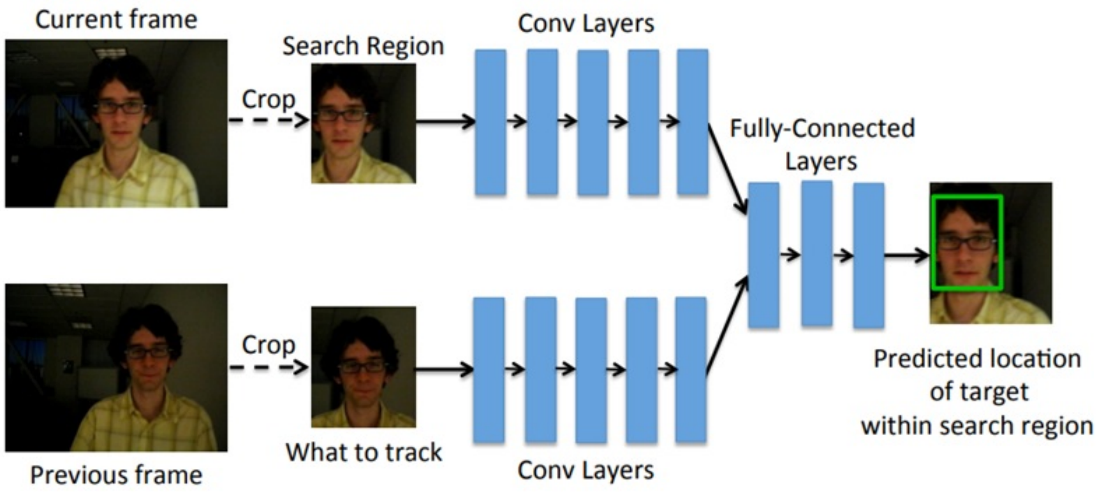


Figure 3.4: GOTURN Architecture (Held *et al.*, 2016)

CHAPTER 4

Tracking and Face Re-identification using Deep Networks

The previous chapters gave a rough overview of the technology that was used in our project. We have made improvements that make the system better at tracking faces. We highlight them now.

4.1 From Generic to Face

As we know that GOTURN is a generic object tracker which can track any object given to it to track. For our application we wanted a fairly specific face tracker which when supplied with face bounding boxes from the detection algorithm was able to track them in subsequent frames.

4.1.1 Training on faces

The original GOTURN algorithm uses the first five layers of the CaffeNet to extract features of the object and then passes the features extracted from the previous and the current frame to a fully connected network of 3 layers each with 4096 nodes. Finally the last layer is connected to the output layer with 4 nodes for the new bounding box.
The first logical thing to make it face specific is to train the network on face dataset. What better than the already described WIDER dataset and the VGG Face Dataset. To teach our network to prefer small motions to large motions, we augment our training set with random crops drawn from the Laplace distributions. That is we move the faces in the image by some amount and also change the size of the bounding box a little to train from images. This new network is now called FTURN (Face Tracking Using Regression Networks)
But this was generic object tracker with no knowledge of how a face looks like in

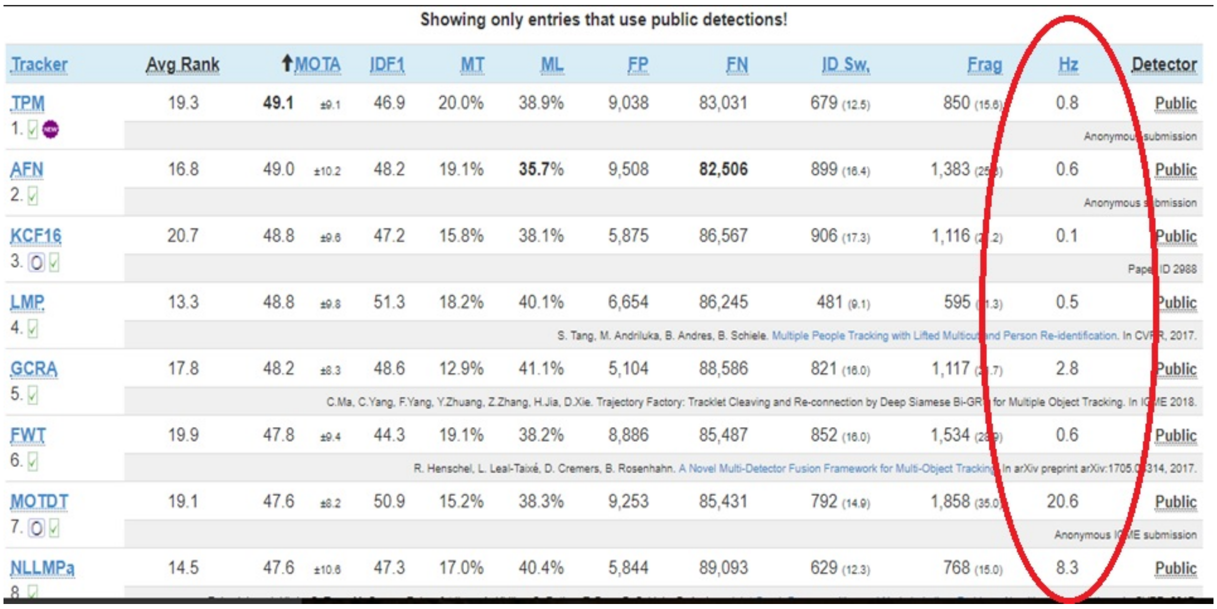
general settings. We now needed to provide it with a representation of the face from the previous frame so that it can get an idea about general face features like texture and color which it needs to find in the current frame.

4.1.2 Adding general face features

For providing our tracker with the knowledge of how a face looks like we added the feature detection layers of the face detection module from the SSH detection network to our tracking network. These layers would help to get the features from the image that would be the general face features. These extracted features when concatenated with other features would help our network to be able to track the faces better. We added the layers along with the weights of the SSH layer that was supposed to detect faces that are large sized. This layer has helped the tracker by finding the feature in a frame that distinctly identify a face. The tracker performs much better than the original tracker and also the tracker trained on faces.

4.2 From Single object to Multi-Object Tracking

The next target was to make the tracker track multiple faces so that we can get the profile of many people seeing the ad together and can do a comparative study based on the emotions or soft-biometric features of the crowd.
We know that GOTURN in itself is a single object tracker. We used multi-threading to make multiple objects of the same tracker class. Each of these object tracks a single face in the crowd and multiple of these objects are running at the same time. These trackers also use just a single tracker net and thus help save memory on the GPU.
We are able to track multiple objects in real time at a frame rate of about 30 FPS. It is faster than most trackers participating in the MOT (Multi Object Tracking) Challenge (Milan *et al.*, 2016). Fig 4.1 shows the frame rates (in Hz) of top trackers in the challenge.



Showing only entries that use public detections!

Tracker	Avg Rank	MOTA		IDF1	MT	ML	FP	FN	ID Sw.	Frag	Hz	Detector
TPM 1. ✓ ●	19.3	**49.1**	±9.1	46.9	20.0%	38.9%	9,038	83,031	679 (12.5)	850 (15.6)	0.8	Public
												Anonymou[illegible] submission
AFN 2. ✓	16.8	49.0	±10.2	48.2	19.1%	**35.7%**	9,508	**82,506**	899 (16.4)	1,383 (25[illegible])	0.6	Public
												Anonymous [illegible]bmission
KCF16 3. O ✓	20.7	48.8	±9.6	47.2	15.8%	38.1%	5,875	86,567	906 (17.3)	1,116 ([illegible].2)	0.1	Public
												Pape[illegible] ID 2988
LMP 4. ✓	13.3	48.8	±9.8	51.3	18.2%	40.1%	6,654	86,245	481 (9.1)	595 ([illegible]1.3)	0.5	Public
												S. Tang, M. Andriluka, B. Andres, B. Schiele. Multiple People Tracking with Lifted Multicu[illegible]nd Person Re-identification. In CV[illegible]R, 2017.
GCRA 5. ✓	17.8	48.2	±8.3	48.6	12.9%	41.1%	5,104	88,586	821 (16.0)	1,117 ([illegible].7)	2.8	Public
												C.Ma, C.Yang, F.Yang, Y.Zhuang, Z.Zhang, H.Jia, D.Xie. Trajectory Factory: Tracklet Cleaving and Re-connection by Deep Siamese Bi-GR[illegible] for Multiple Object Tracking. In I[illegible]ME 2018.
FWT 6. ✓	19.9	47.8	±9.4	44.3	19.1%	38.2%	8,886	85,487	852 (16.0)	1,534 (2[illegible])	0.6	Public
												R. Henschel, L. Leal-Taixé, D. Cremers, B. Rosenhahn. A Novel Multi-Detector Fusion Framework for Multi-Object Tracking. In arXiv preprint arXiv:1705.[illegible]314, 2017.
MOTDT 7. O ✓	19.1	47.6	±8.2	50.9	15.2%	38.3%	9,253	85,431	792 (14.9)	1,858 (35.0)	20.6	Public
												Anonymous I[illegible]ME submission
NLLMPa 8. ✓	14.5	47.6	±10.6	47.3	17.0%	40.4%	5,844	89,093	629 (12.3)	768 (15.0)	8.3	Public

Figure 4.1: MOT Challenge entries

4.3 ResNet

Recent papers and competition winners have used ResNet architecture for various purposes related to object classification and detection.

Residual Network developed by Kaiming He et al. (He *et al.*, 2015) was the winner of ILSVRC 2015. It features special skip connections and a heavy use of batch normalization. The architecture is also missing fully connected layers at the end of the network. ResNets are currently by far state of the art Convolutional Neural Network models and are the default choice for using ConvNets in practice.

The author of the paper (Wen *et al.*, 2016) has trained ResNet architecture to extract face feature. We removed the GOTURN feature extraction CNN and replaced it with the CNN layers of the DeepFace architecture described in the above paper (Wen *et al.*, 2016). The network was then again trained to fine tune the weights of the fully connected layers for the ResNet feature extraction network. The ResNet architecture allows the algorithm to decide the size of the network to best represent the images in least amount of parameters. This helps save memory on GPU to save these weights.

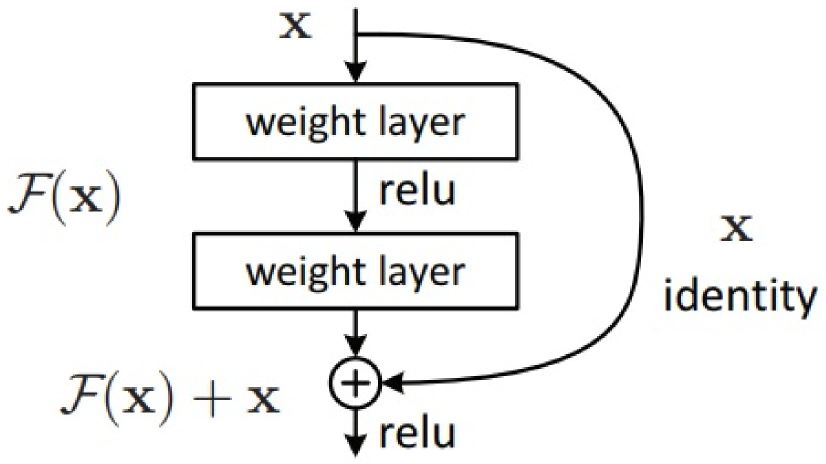


Figure 4.2: Single ResNet unit

It was observed that the tracking bounding boxes around the face were generally more stable in the sense that the size was not changing when the size of the face did not change in the frame. It was also observed that for the same video the memory occupied was 6% less than that occupied by the network with SSH detection network introduced.

4.4 Face Verification

We need to know when our tracker fails to track a face assigned to it. To do that we introduce a similarity layer which would compare the features extracted from the previous frame and current frame and give out a number denoting how similar are the faces being tracked across the frames. The similarity measure used is the cosine similarity and based on experimentations it was noted that cosine similarity below 0.20 was the output if the objects were not similar. This was thus set as the threshold to re-initialize the detection algorithm. The fig 4.3 shows the part of network where the similarity module was added.

4.5 Person Re-Identification

We trigger the detection algorithm SSH when we either lose track of a person or every 600th frame to check if any new viewer has joined the crowd. After detection we need

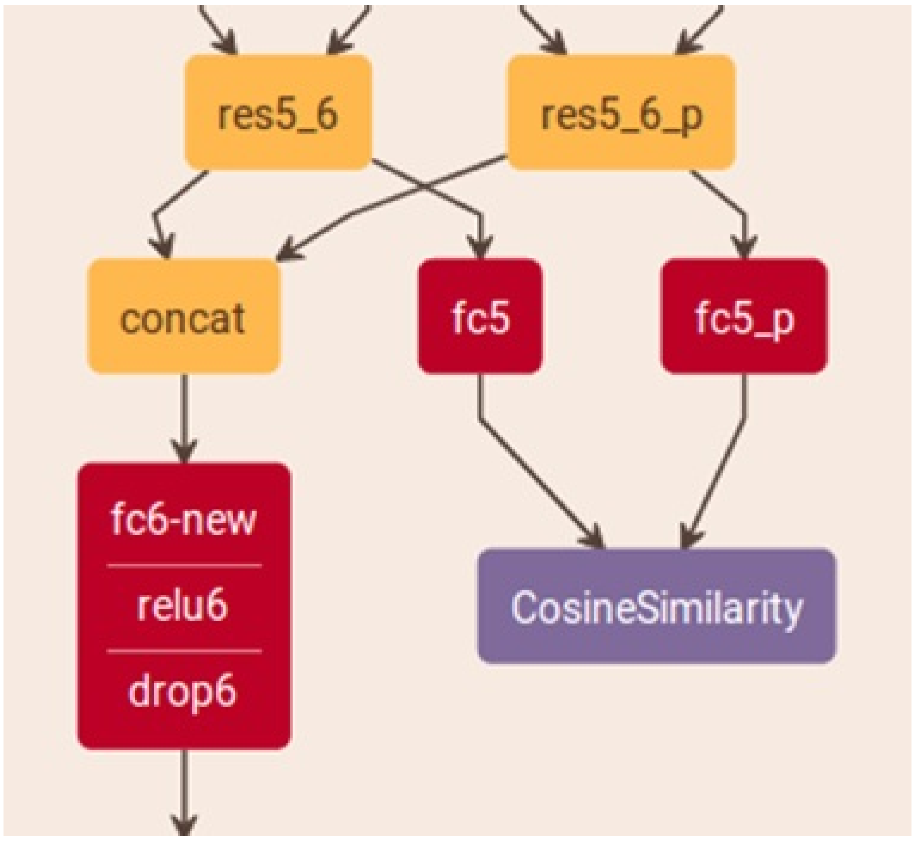


Figure 4.3: Module for finding Cosine Similarity

to assign the same ids to the already detected and tracked faces if they are still there. To do this we use location information and the extracted face features to compare the new detections with the faces that were being tracked. This is Person Re-Identification (Farinella *et al.*, 2014)

We assume that the person at a certain location in the frame will not move far during the detection time and hence we match the features of people in a certain area to people in nearby areas. This helps in continuation of data collected being associated to a single profile for a particular person.

The video shocasing the same has been uploaded at this Link

CHAPTER 5

Results and Conclusions

Below are the links to the videos with tracking done using different changes made to the original tracker.

- Original GOTURN
- FTURN
- SSH layers introduced
- ResNet

Based on multiple experimentations we conclude that the tracker with the knowledge of how a general face look like either by using the layers of detection network or by using the feature extraction layers which are trained specifically to detect faces like the ResNet layers of DeepFace, is able to track faces in a more robust way and is affected less by partial occlusion of the target. Even when the target is fully occluded for a short period of time, our tracker is able to regain it when it appears again. We also notice that the ResNet tracker is generally more stable in tracking the subject though it can mis-track a target in full occlusion conditions, but regains the target when it reappears.

We also show the PoC of Re-Identification with 2 subjects and the video demonstrating it is added above. We are able to recognize the person when he reappears in the frame and assign the same ID as before. This will help us to assign the features to the same profile for a particular person.

REFERENCES

1. *CS231n Convolutional Neural Networks for Visual Recognition* (retrieved: 2018-05-08). URL `http://cs231n.github.io/convolutional-networks/`.

2. **Farinella, G. M.**, **G. Farioli**, **S. Battiato**, **S. Leonardi**, and **G. Gallo**, Face re-identification for digital signage applications. *In* **C. Distante**, **S. Battiato**, and **A. Cavallaro** (eds.), *Video Analytics for Audience Measurement*. Springer International Publishing, Cham, 2014. ISBN 978-3-319-12811-5.

3. **He, K.**, **X. Zhang**, **S. Ren**, and **J. Sun** (2015). Deep residual learning for image recognition. *CoRR*, **abs/1512.03385**. URL `http://arxiv.org/abs/1512.03385`.

4. **Held, D.**, **S. Thrun**, and **S. Savarese**, Learning to track at 100 fps with deep regression networks. *In European Conference Computer Vision (ECCV)*. 2016.

5. **Jia, Y.**, **E. Shelhamer**, **J. Donahue**, **S. Karayev**, **J. Long**, **R. Girshick**, **S. Guadarrama**, and **T. Darrell** (2014). Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*.

6. **Milan, A.**, **L. Leal-Taixé**, **I. Reid**, **S. Roth**, and **K. Schindler** (2016). MOT16: A benchmark for multi-object tracking. *arXiv:1603.00831 [cs]*. URL `http://arxiv.org/abs/1603.00831`. ArXiv: 1603.00831.

7. **Najibi, M.**, **P. Samangouei**, **R. Chellappa**, and **L. Davis**, SSH: Single stage headless face detector. *In The IEEE International Conference on Computer Vision (ICCV)*. 2017.

8. **Parkhi, O. M.**, **A. Vedaldi**, and **A. Zisserman**, Deep face recognition. *In British Machine Vision Conference*. 2015.

9. **Redmon, J.** and **A. Farhadi** (2016). Yolo9000: Better, faster, stronger. *arXiv preprint arXiv:1612.08242*.

10. **Wen, Y.**, **K. Zhang**, **Z. Li**, and **Y. Qiao**, A discriminative feature learning approach for deep face recognition. *In European Conference on Computer Vision*. Springer, 2016.

11. **Yang, S.**, **P. Luo**, **C. C. Loy**, and **X. Tang**, Wider face: A face detection benchmark. *In IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.