Evaluation of Face Detectors and Feature Association Metrics for Real-time Multi-Face Tracking

Jianzhou Wang

Thesis submitted to the University of Ottawa
in partial Fulfillment of the requirements for the
M.A.Sc. degree in
Electrical and Computer Engineering

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Abstract

Video annotation, control of camera direction, labelling and other tasks can benefit from online visual multi-face tracking. Given the availability of high quality general purpose detectors and tracking-by-detection frameworks, we develop a multi-face tracker and comparatively evaluate its components. In this thesis, we train common object detectors on large databases of faces to understand how well these detectors can perform, specifically on faces. We evaluate different face association methods and appearance metrics with different classifier loss functions to track detected faces across frames. We find that while online tracking based on combining state-of-the-art methods can lead to high-quality tracking results, there is still a large gap between offline and online methods. We develop a multi-tracking system in order to achieve an online and real-time standard, one that can track most of the faces in unconstrained settings.

Declaration

In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of University of Ottawa's products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to `http://www.ieee.org/publications_standards/publications/rights/rights_link.html` to learn how to obtain a License from RightsLink.

Acknowledgements

First of all, I would like to express my great gratitude to my research supervisor also being my thesis adviser, Professor Jochen Lang, for providing me with excellent research environment, valuable directions and delicate guidance through my master studies. His diligence, concentration and great passion towards to scientific research work inspires me to carry all my effort to all the journeys to come. Without his encouragement and solid support, I could not go any further in this tremendous research field. I also would like to thank Muye Jiang. His generous support is significantly improved my computer skills and research work.

Dedication

I would like to thank my parents who support me mentally and financially since throw back to 9 years ago when the first time came to Canada in 2011, I was in the dark, unprepared, and I would not achieve anything without their support.

List of Abbreviations

- CNN: Convolutional Neural Networks
- MTL: Multi-Task Learning
- NMS: Non-Maximum Suppression
- MOTA: Multiple Object Tracker Accuracy
- MOTP: Multiple Object Tracking Precision
- IOU: Intersection Over Union
- SVM: Support Vector Machine
- NMS: Non-Maximum Suppression
- ELU: Exponential Linear Unit
- ReLu: Rectified Linear Unit
- SGD: Stochastic Gradient Decent
- MTCNN: Multi-Task Convolutional Neural Network
- SSD: Single Shot Detector
- RNN: Recurrent Neural Networks
- R-CNN: Region-based Convolutional Neural Network
- R-FCN: Region-based Fully Convolutional Networks
- RPN: Region Proposal Networks
- ResNet: Residual Network

- STN: Spatial Transform Network
- DA: Data Association
- HA: Hungarian Algorithm
- NNDA: Nearest Neighbor Data Association
- JPDA: Joint Probabilistic Data Association
- SORT: Simple Online and Realtime Tracking
- MHT: Multiple Hypothesis Tracking
- ACF: Aggregate Channel Filter
- XML: Extensible Markup Language
- CSV: Comma-Separated Values
- AP: Average Precision

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Facial detection and tracking systems are well-researched in the field of image processing and machine vision. Such techniques utilize deep learning [80]; with applications in mobile face recognition systems [1] [57] [50], security surveillance [70] [96] [5] [47], human behavior analysis [60] [95], etc. Due to recent progress in multi-object detection, tracking-by-detection has become the most popular method for multi-face tracking. Two state-of-the-arts multi-face trackers: ADF tracker [93] and Lin et al. [49] have demonstrated good performance in unconstrained videos. A simple online and real-time tracking method based on deep association metrics (Deep SORT) [83] works well for pedestrian tracking with the cosine softmax classifier [82]. In this thesis, we adapt the Deep SORT for multi-face tracking by changing to face feature metrics and by substituting the cosine with the angular spft,ax classifier to distinguish different faces. The accurate detection of faces is a crucial step for tracking. Therefore, we integrate and test a dedicated face detector [91] and common object detectors [51] [66] [16] with different feature extractors [74] [27] [32] for the multi-face tracking input.

1.1 Background Review

Traditionally, face detection and tracking are based on image pixel features such as Texture Descriptors, Edge Detection, and Histogram of Oriented Gradients. Human face detection and tracking in the real world are usually in unconstrained environments due to high variability of angles, head poses, ages, facial expressions occlusions and illumination conditions.

A typical face tracking system follows the steps listed below:

- Face detection methods find the positions of the faces in images and return the coordinates (pixels locations) of the bounding box for each one of the images.
- Face alignment identifies the geometric structure of faces in images, and gives the location and size of the face through a set of reference points.
- A face representation is found by transforming pixel values of a face image into a compact and discriminative feature vector. This process makes all the images from a face map to the discriminative feature vector.
- At the face matching stage, two feature vectors from face representations are compared to produce a similarity score (the likelihood to find which person's face belongs to).

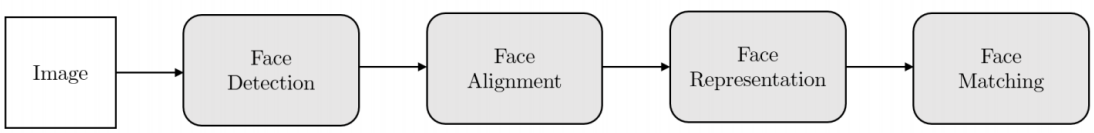

Figure 1.1: Face Recognition Blocks Diagram [76]

Many different face detection and tracking detection have been developed. The Viola-Jones face detector [78] has been a seminal work in face detection proposed in 2001. Many

relevant reviews have been conducted, e.g., Zhao et al. [94] reviewed a list of face recognition systems. Yang et al. [86] listed a dozen of common visual tracking systems and Chrysos et al. [15] presented deformable face tracking pipeline. With the developments of deep learning methods, these are now commonly used in face detection and tracking. Deep learning based methods of face detection and tracking provide more accurate and efficient approaches. Deep learning in face detection uses multiple layers to extract higher-level features from the raw input. This method requires the use of data to train the desired subject, rather than design specialized features that are robust to different types of intra-class variations (i.e., illumination, pose, facial expression, age, etc). This way, facial features and facial landmarks can be identified through deep learning. Multiple commonly used datasets, i.e., PASCAL [72] [88], WIDERFACE [62] [23], FDDB [61] [13] are open for public use. Many state-of-the-art detectors have been pre-trained on COCO [3], a large scale image dataset. Transfer learning is a method where a model developed for a task can be reused as the knowledge for a model on a second task. We can apply transfer learning to these pre-trained detectors with face datasets to obtain multi-face detectors.

Face tracking is the processing of locating moving faces over time in videos. Multi-face tracking differs from single face tracking because the appearance alone is not enough to track multi-faces across frames. Multi-face is not a binary decision. A single face in view such as a zoom call is a simple binary scenario while multi-face tracking in unconstrained videos can be a complex scenario which brings more challenge to track. Single face tracking as in a video conference has the challenge of detecting all the facial landmarks with their prediction and verification precisely [56]. Single face tracking in varied and unconstrained environments are usually involved with noises from backgrounds. The objective is to indicate whether or not the person's face is in the rest of the frames. This is the re-identification problem [92]. However, multi-face tracking, similar to the multi-object tracking such as pedestrian tracking is to algorithmically associate the detections from faces in each frame [44].

Multi-face tracking can use data association techniques to link motions of faces in previous frames to faces in the current frame. Recent methods are applied with deep features on faces through tracking-by-learning detection [36], which can be improved by using deep convolutional neural networks.

1.2 Motivation of this thesis

Two state-of-the-art multi-face offline trackers [93] [49] have achieved very good results in unconstrained videos when benchmarked with the Multi-Object-Tracking (MOT) evaluation metrics [59]. However, these two state-of-the art multi-face offline trackers process video frames offline for tracking, often taking a significant amount of time to complete. These more evolved methods cannot be performed in real-time on video streams. Online tracking versus offline tracking differs in whether or not observations from future frames are utilized when handling the current frame.

The motivation of our work is to search for suitable detectors as the online tracking input and to understand the gap between online and offline methods for tracking-by-detection when standard blocks for detection and data association are adapted. Our goal is to determine if high accuracy results can be achieved in real-time when performing online multi-face tracking. In addition, we are seeking how to classify multi-faces from the multi-face tracking matching cascade with different feature classifiers.

1.3 Our contributions

We have the following contributions in this thesis:

1. We train common object detectors and evaluate them for faces as multi-face detectors through transfer learning and compare them with the dedicated MTCNN [91] to select

suitable multi-face detectors.

2. We implement different classifier loss functions to identify functions well suited for use in feature distance metrics of faces. Then we further develop and adapt an online tracker on faces then compare it to state-of-the-art multi-face offline trackers and online trackers, and give recommendations for real-time online multi-face tracking.

3. We create a new dataset for a new multi-face tracking benchmark and evaluate our online tracker adaptation, then compare with another online tracker.

1.4 Thesis Structure

This thesis is organized as follows:

- Chapter 2 introduces basic concepts such as Neurons, Neural Networks, Convolutional Layers, Activation Function, and training parameters of CNNs.

- Chapter 3 describes the related work of multi-face detection: MTCNN, and three common object detectors: SSD, Faster R-CNN, and R-FCN. Chapter 3 also introduces Data Association techniques for tracking such as Kalman Filter and Hungarian Algorithm, multi-face trackers such as Simple SORT and Deep SORT, and Classifier Loss Functions.

- Chapter 4 presents the proposed approach of multi-face detection and tracking system.

- Chapter 5 demonstrates the methodology, experiment setup and evaluation metrics of multi-face detection tasks and multi-face tracking tasks.

- Chapter 6 presents the results and analysis of our multi-face detection and tracking systems.

- Chapter 7 concludes the thesis work.

Chapter 2

Background

Face detection has used machine learning for dozens of years. Our tasks include the classification of faces and finding their locations. It is analogous to image detection in which the image of a face is matched pixel by pixel, then features can be extracted and used in comparing possibly matching faces. Traditional facial detection algorithms focus on the detection of frontal human faces. A lot of tedious work needs to be done before face extractions, such as reducing the lighting effect and noise as well as brightness adjustment. In order to avoid this tedious work and to further improve the detection accuracy, deep learning with cascaded convolutional neural network approaches can achieve impressive performance on this task [91].

For multi-face tracking, data association links the correlated faces and connects a series of same or similar faces. It gives all possible detections in a frame that it determines to track. However, deep feature classification has enabled to achieve drastically increased performance on tracking different faces. In particular, Simple Online and Realtime Tracking with Deep Association Metric (Deep SORT) [83] will be investigated in thesis.

2.1 Convolutional Neural Networks

Convolutional neural networks (CNNs) emerged from the study of the brain's visual cortex, and they have been used in image recognition since the 1980s [77]. A convolution is a mathematical operation that can be understood as sliding one function over another and measuring the summation of their point-wise multiplication. The convolution operation in 2-D images can be implemented with a square filter kernel that is moved over the image from pixel to pixel. Convolutional Neural Networks are an effective approach to extract object features for classifying objects. Neural Networks in machine learning are inspired by biological neural network architectures, where individual cortical neurons respond to stimuli [42]. Each neuron in ML receives inputs, performs a dot product and optionally follows it with a non-linear operation. The entire network has building blocks consisting of Convolutional Layers, Fully Connected Layers(FCL) and Pooling Layers to output a single differentiable score function which expresses a class scores from the input of raw image pixels.

A Convolutional Neural Network (CNN) takes an input image, assigns learnable weights and biases to the objects in the image and differentiates one from the other. Figure 2.1 presents a handwritten Arabic numeral being classified by a Convolutional Neural Network architecture.

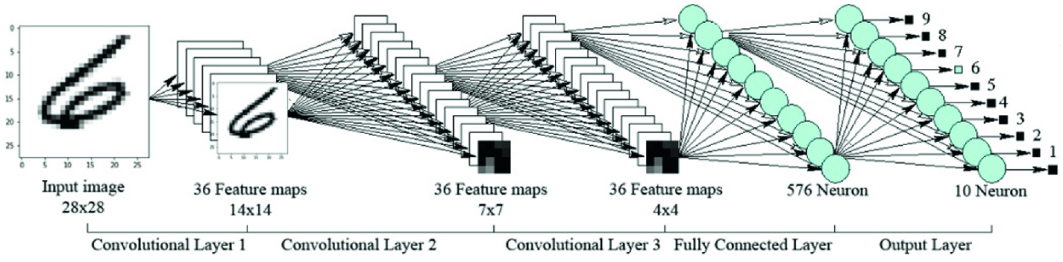

Figure 2.1: (©2018 IEEE)A CNN Sequence to Classify Handwritten Digits [69]

2.1.1 Neuron

Neurons are the basic units in a neural network. In machine learning, they are mathematical functions with one or multiple inputs x_i and an output y. Each neural input x_i is multiplied by its own weight w_i. The additional term – bias b, is added to the sum of multiplication of each input x_i by its corresponding weight w_i. The activation function f has the role to initialize the neural function and introduce a non-linearity effect. The neuron with k inputs can be formulated as demonstrated in the Figure 2.2.

$$y = f\left(\sum_{i=1}^{k} w_i \ * x_i + b\right) \tag{2.1}$$

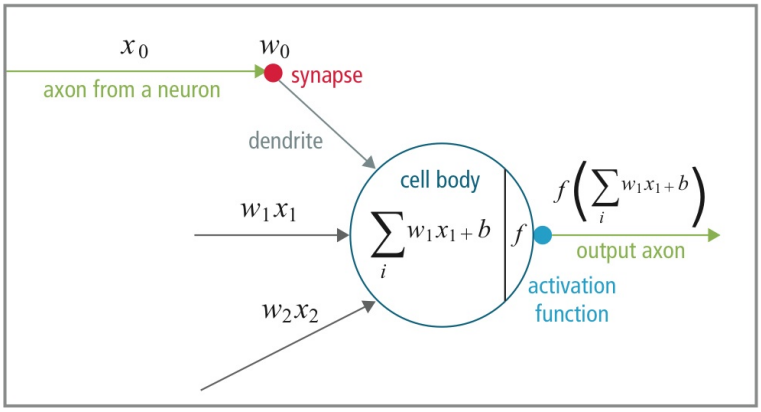

Figure 2.2: (©2016 Li)Neuron Structure [45]

2.1.2 Neural Networks

A neural network forms a system that connects the artificial "neurons" network that passes the information between each other within layers. The weights are updated during the training process, so the image feature can be correctly detected in a properly trained neural network. A neural network consists of multiple layers of feature neurons. For example, a neural network function consists of four layers f_1, f_2, f_3, f_4. Thus the output function can be represented as

$$f(x) = f_4(f_3(f_2(f_1))) \tag{2.2}$$

In this system, the neuron in the first layer f_1 transmits the input to the second layer f_2. Intuitively, the third layer f_3 transmits to the last layer f_4 (the output layer). The layer between the input and the output, such as f_2 and f_3, are called hidden layers since they are not shown at the two ends the network, i.e., input and output. Many layers construct a deeper network for training and learning, named Deep Learning. A two hidden layers neural network is shown below:

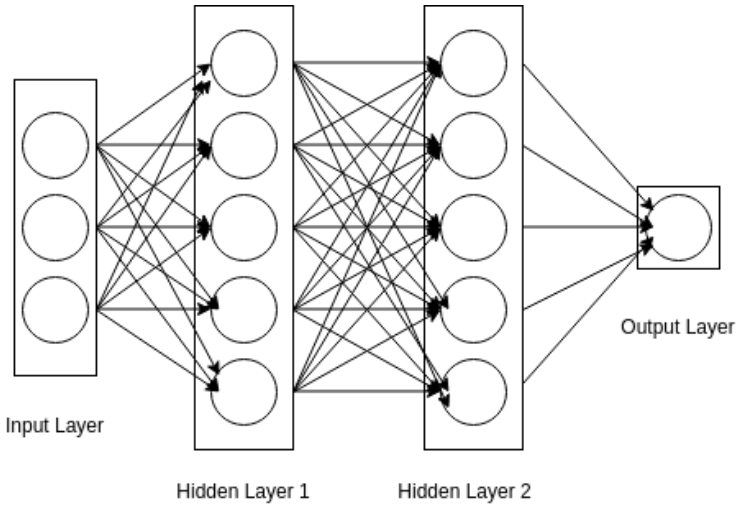

Figure 2.3: A Two Hidden Layers Neural Network

2.1.3 Convolutional Neural Networks

Convolutional neural network architecture typically consists of convolutional layers, pooling layers, fully connected layers and dropout layers. When the face image inputs into CNNs, features of face images are presented in multiple vectors. When CNNs receive a single vector and transform it through a series of hidden layers, also known as convolutional layers, each hidden layer has multiple neurons. Each neuron is fully connected to all neurons in the previous layer but independently with other layer functions.

Convolutional Layer

As neurons are introduced in Section 2.1.1, a neural network layer can be calculated as

$$y = f\left(W * x + b\right), \tag{2.3}$$

where the W, x, b and y are the metrics of weights, convolutional inputs, bias, and output, respectively. The key ingredient of a CNN is the convolutional layer, in which a sliding window calculates the result of the input layer and each convolutional layer. In image processing and pattern recognition, the convolutional kernel value is the weight corresponding to the local pixel values. In convolutional layer, convolution operation slides the kernel across the image. Summing the multiplication of convolutional kernel values and their respective pixel values with the addition of bias value give the result of the convolutional layer output. Figure 2.4 represents the convolutional layer process of 2 x 2 kernel on a 3 x 3 image to produce a 2 x 2 output result.

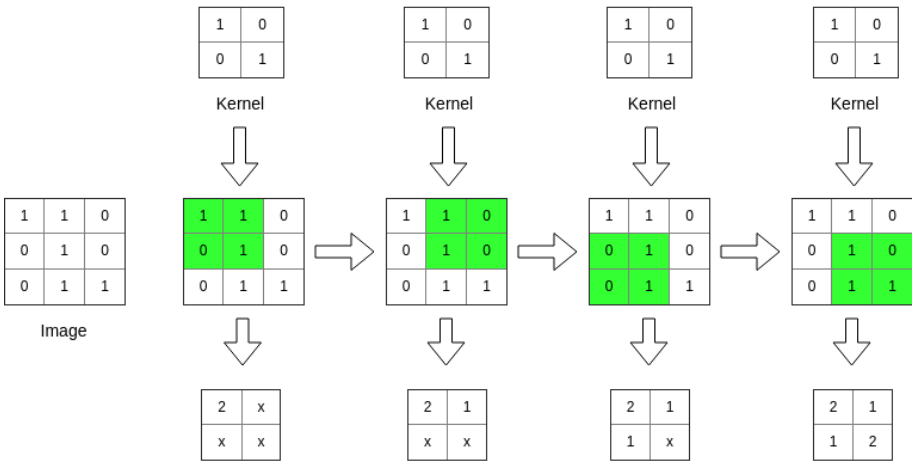

Figure 2.4: A Convolutional Layer

Pooling Layer

The pooling layer is another building block added after convolutional layers. It down-

samples feature maps by summarizing and combining the outputs of neurons at one layer into a single neuron to the next layer. This process reduces computational cost and avoids overfitting which occurs when the training set is simple or the function has many parameters to fit. Max-pooling and Mean-pooling are two pooling methods commonly used in CNN. Max-pooling selects the maximum value in the pooling window, and mean-pooling takes the average of the summation of all the values in the pooling window.

Fully Connected Layer

The fully connected layer, also known as the inner product layer, calculates the class scores corresponding to output categories. It acts as a classifier, and it is put in the last few layers to give a result of the class.

Dropout Layer

The dropout layer is a regularization technique created by G.E.Hinton [30]. Dropout layer can randomly drop the hidden or visible layers to solve the problem of over-fitting by preventing a complex model to closely model the training data.

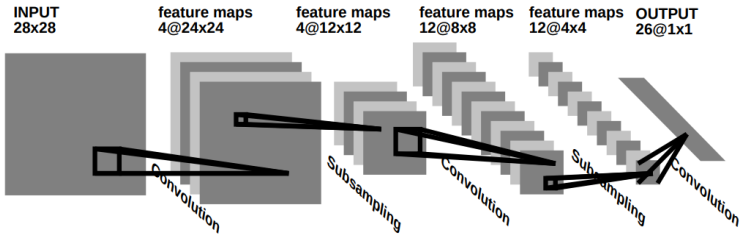

Figure 2.5: (©1995 IEEE) Convolutional Neural Network Architecture [43]

2.1.4 Activation Function

Activation functions decide whether a neuron is activated. Activation functions approximate more complex functions by introducing non-linearity, e.g., Sigmoid and ReLu.

The Sigmoid Function is also known as Logistic Function. It is defined as

$$Sigmoid\,(x) = \frac{1}{1 + e^{-x}} \tag{2.4}$$

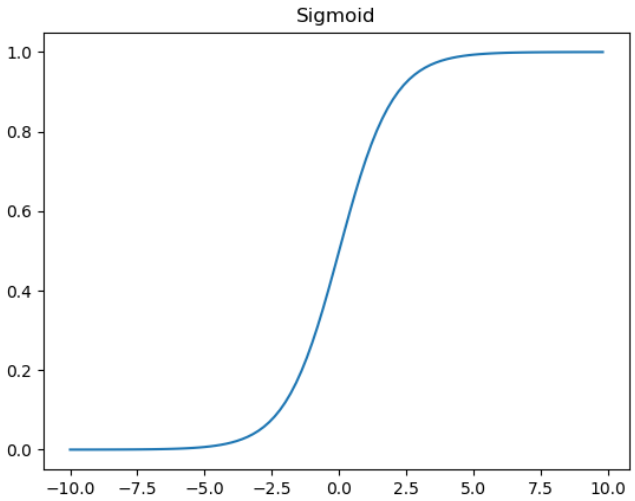

Figure 2.6: Activation Function: Sigmoid

This is used in neural networks to predict yes (1) or no (0) to activate it or deactivate it. If the x approaches negative infinity, the output would result in 0 to deactivate the function. Likewise, if the x approaches positive infinity, the output would result in 1 to activate the function.

Rectified Linear Unit (ReLU) is also known as a ramp function because it is analogous to half-wave rectification in electronic circuit analysis. It also has been commonly used in deep learning models. It is defined as

$$ReLU\,(x) = max\,(0, x) \tag{2.5}$$

ReLu function returns 0 if it receives any negative input but for any positive value x, it returns that value to activate the function. It has zero gradient for $x < 0$, and is non-zero centered, non differentiable at x = 0. It allows the model to calculate non-linearity and

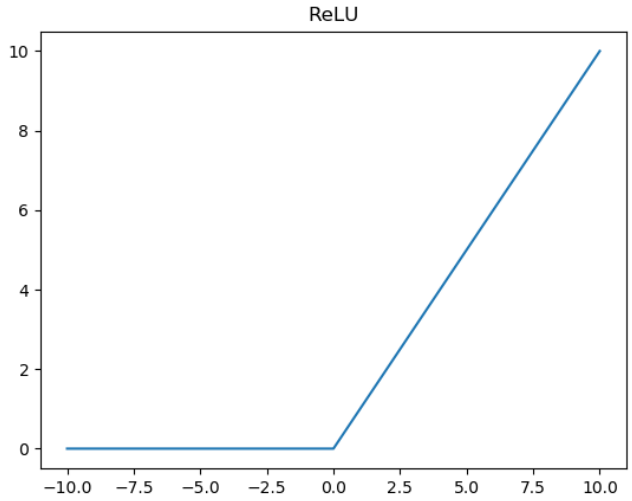

Figure 2.7: Activation Functions: ReLU

perform interactions well.

2.2 Machine Learning

Machine learning algorithms improve itself based on experience, i.e., they learn from data input to the algorithm. There are three general categories in machine learning based on labels and feedback available to the system: (1) Supervised Learning, (2) Unsupervised Learning, and (3) Reinforcement Learning.

In supervised learning, the training data used into the algorithm includes the desired solutions, named labels. Classification and regression are often utilized. Classical machine learning algorithms for supervised learning include Support Vector Machines, Decision Trees, Random Forests, Logistic Regression, and Gradient Boosting.

In unsupervised learning, the system is learning without labels nor feedback through the learning process. For example, the clustering task is one method that groups similar data together. Another related task in unsupervised learning is dimensionality reduction

which aims to simplify complex data. Moreover, the dimensionality reduction algorithm merges data into one feature which represents a variety of possible influences, also known as feature extraction.

In reinforcement learning, the system enables an agent to take action to reach the maximum reward in a dynamic setting. It can be distinguished from supervised learning because it reinforces the signal to provide feedback and in turn, rewards the process.

2.3 Hyper-Parameters

In machine learning, hyper-parameters are defined and set before the learning process, so they are not obtained during the learning process. When training a machine learning algorithm, it requires a careful selection of hyper-parameters since those values are used to control the learning process. For example, validation and test losses may reflect overfitting and underfitting, while selecting well-suited hyper-parameters values and tuning hyper-parameters help to reduce the influence from these two issues. There are several hyper-parameters such as Learning Rate, Batch Size, Weight Decay, Epoch and Step important to the implementation.

- Learning rate determines the speed of the learning progress. Its schedule seeks to adjust the learning rate for updating the weight during training according to a pre-defined schedule. Small learning rates cause the training progress to be slow, and large learning rates train the model quickly but risk missing the global minimum. However, learning rate decay is scheduled to decrease the learning rate by a small value and can be a trade-off solution to ensure fast learning from the beginning to a near flat region without missing the global minimum.

- Batch size refers to the number of training examples utilized in one iteration. Different size of the batch influences the optimization and speed of the training. If the

dataset is large, using the entire dataset for training over one iteration may cause the exploration of the training process. The different size of the batch influences the optimization and speed of the training. The mini-batch mode has a batch size greater than one but less than the total dataset size.

- Weight decay, also known as $l2$ regularization, is used after each update to reduce the weight by multiplying the differentiable factor which is less than 1. This procedure prevents the over-fitting due to the growth of weights being large.

- Epoch is equivalent to the size of the dataset (number of images). The number of epoch equals the number of iterations of training the entire dataset. The decay rate and learning rate have the following relation: Decay Rate = Learning Rate/Epochs.

- Step is calculated by

$$Step = \frac{SampleNums * Epoch}{BatchSize} \tag{2.6}$$

where $SampleNums$ is the numbers of samples to be trained. For example, if the numbers of images are 100,000, the batch size is 200, and the epoch is 1, then the step = 500. The weight changes for every step.

Chapter 3

Related Work

A multi-face detection and tracking system relies on two sub-systems: (1) Multi-face Detection System; (2) Multi-face Tracking System. Many researchers [51] [19] [64] [91] [62] characterize multi-object detection systems as models that achieve good speed and accuracy. Here, we need to choose suitable detectors trained with face datasets as tracking inputs for our multi-face tracking system. On the other hand, many multi-object trackers [36] [7] [28] [14] [2] [38] [24] [75] [17] [89] [39] including two state-of-the-art multi-face offline trackers [93] [49] show good performance in unconstrained videos.

In this chapter, we present a face detector using Multi-task Cascaded Convolutional Networks [91] and three common object detectors [51] [66] [16] in Section 3.1. which are the prerequisites for our multi-face tracking work. In Section 3.2., we present the deep association matching cascade by Wojke et al. [82] that can work with different classifier loss functions.

3.1 Multi-face Detection Systems

A multi-face detector requires a combination of face classifiers and bounding box regression. For each face, the outputs of the face detection are classification scores, and the bounding

box regression gives outputs of locations of faces and their landmarks. In the past, the Viola-Jones face detector [78] was proposed by Paul Viola and Michael Jones, and it was the first face detection framework that demonstrated impressive performance and detection speed which was approximately 100 times faster than any face detection methods back to 2001. Viola-Jones detector built a simple and efficient classifier based on AdaBoost learning algorithm and combined these classifiers in a "cascade" form to enhance the ability of removal of background regions. Zhao et al [94] have listed video-based face recognition systems and presented detailed descriptions of representative methods in each tracking category. Zhao et al [94] also emphasized face recognition systems that are automatic by tackling problems such as localization of a face in video frames and extraction of features of overall faces and their landmarks as well as outstanding classifiers have been made for successful face recognition.

The Joint Face Detector using Multi-Task Cascaded Convolutional Networks (MTCNN) [91] a three-stages multi-face detector is one of the implementations that we use in our multi-face detection. Three existing common object detectors are also presented. Using transfer learning of multi-face detectors from pre-trained multi-object detector models on face dataset obtains multi-face detectors. One of the detectors, Single Shot Multibox Detector (SSD) [51], is a one-stage detector that uses deep convolutional neural networks to process both region proposal and region classification It is compared with other two two-stages detectors: Faster R-CNN [66] and Region-based Fully Convolutional Networks (R-FCN) [16]. Faster R-CNN consists of the Region Proposal Networks to locate object regions, followed by classification layer and refined bounding box layers. Whereas Faster R-CNN, R-FCN shares feature information on location score maps, followed by Fully Convolutional Region Proposal Networks to generate regions of interests, then using softmax for classification of the regions.

3.1.1 Region Proposals

In multi-object detection, one of the tasks is to classify objects. Another task is to locate the object and draw the bounding box in the image. The Region Proposals, also known as Region of Interests, create the candidate boxes. This can be done by Region Proposal Networks (RPN) that learn the proposals from feature maps. There are three main steps of the RPN: The first step is to generate three to nine anchor boxes based on every anchor point on the feature map. The second step is to classify each anchor box, whether it is foreground or background. The last step is to learn anchor boxes offset values to fit them for objects.

Non-Maximum Suppression

Non-Maximum Suppression (NMS) is a technique commonly used in object detection tasks and aims to filter redundant detection results [79]. NMS is a multi-dimensional principle used to filter out overlapping results and has been used in many computer vision optimizations. For example, when NMS is applied to a SSD model to obtain the optimized bounding box location, the first step is for a set of bounding box list B with the score S, choose the biggest score bounding box M, and move it out from list B and add to the result R. Secondly, if the Intersection over Union (IOU) of rest of bounding boxes in B is bigger than the threshold value (normally 0.30-0.50), then move it to result in R. Third step is to repeat the procedure until list B is empty.

Figure 3.1 shows that we have 3 bounding boxes in or around the person shown before the processing. Based on the probability scores from low to high, we have A (the yellow rectangle bounding box), B (the yellow square bounding box) and C (the green rectangle box bounding box).

1. From the highest score C, and check A, B and C's IOU whether it is bigger than the threshold or not ;

Figure 3.1: (©2017 IEEE) Non-Maximum Suppression [31]

2. If A's IOU with C is bigger than threshold value, then remove bounding box A; keep the bounding box C.

3. Repeat second step: Compare the B's IOU with C to check the threshold value.

4. Finally, we can obtain the highest score among A, B, and C that was not removed.

3.1.2 Cascaded Convolutional Neural Networks in Face Detection

We introduce the cascaded convolutional neural networks by Li et al. [46] for a clear explanation of the proposed detectors. The overall network shows how the face detection bounding boxes labeled in green squares are reduced and calibrated from stage to stage in the detector, finally obtain exact one bounding box for each face.

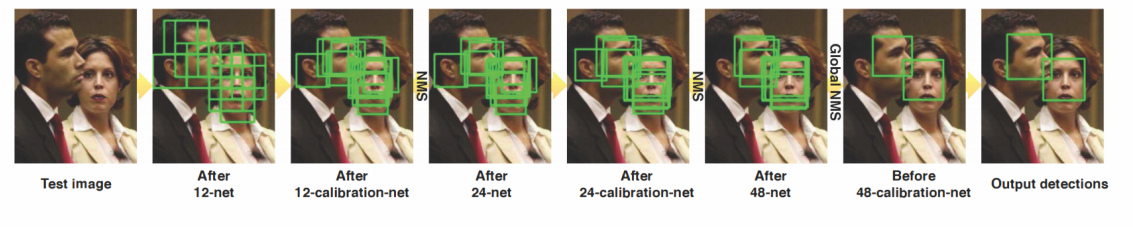

Figure 3.2: (©2015 IEEE) Cascade Convolutional Neural Networks [46]

From Figure 3.2, the 12-net CNN scans the entire image with different bounding box scales to reject a majority of the detection windows. After that, the 12-calibration-net scans the previous detection windows one-by-one to adjust size and location. NMS is applied to remove strongly overlapping detection windows. Similar procedures for 24-net stage and 48-net stage are used to obtain the final output which is only one optimized bounding box on each face.

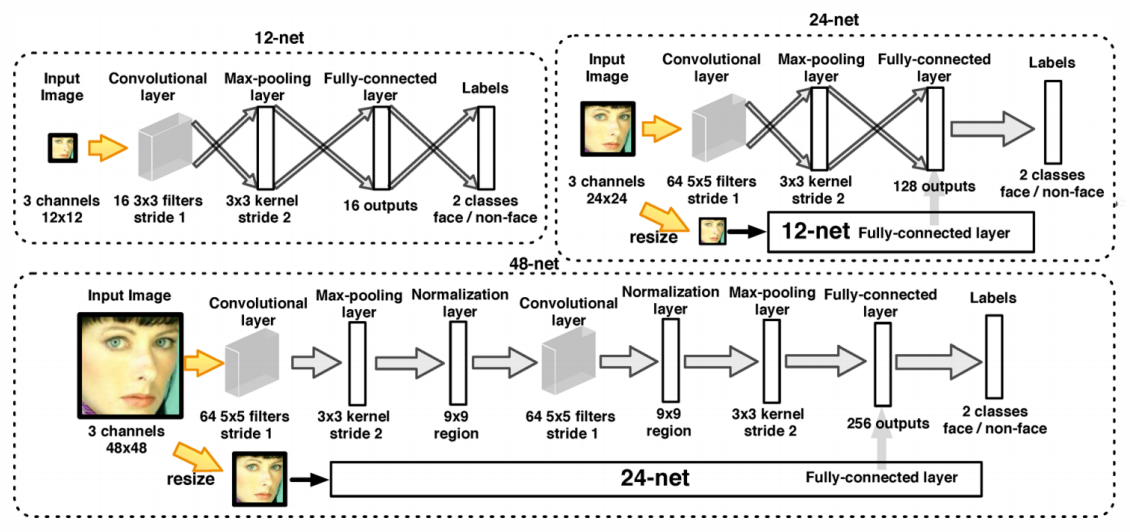

Figure 3.3: (©2015 IEEE) CNN Structures of 12-Net, 24-Net and 48-Net [46]

3.1.3 Multi-Task Cascaded Convolutional Neural Networks(MTCNN)

The MTCNN by Zhang et al. [91] based framework performs in real-time, and it uses simple CNNs connected in cascade form for joint face detection and alignment. This CNN architecture is a modification of the Cascaded Convolutional Neural Networks [46].

The MTCNN consists of three stages:

1. First stage: it produces candidate windows quickly through a shallow CNN. The first fully convolutional network, named Proposal Network (P-Net) generates the candidate windows which are produced through P-Net. The bounding box regress vectors adjust the candidate windows and connect with NMS to filter strongly overlapping candidate windows.

2. Second stage: it refines the windows to reject a majority of candidates that have lower scores by performing calibration with bounding box regression. This stage applies NMS again to filter strongly overlapping candidate bounding boxes. At this stage, only very few candidate windows are left, and this stage is called Refine Network (R-Net).

3. At the final stage, it uses a 48-net CNN to refine the result and output facial landmarks positions. This stage is called the output stage. Besides giving more accurate faces positions, it produces the output with positions of five facial landmarks. This stage is the Output Network (O-Net).

The pipeline of the face detection cascade, and the alignment re-sizes the image to different scales to build an image pyramid. The image pyramid is used as the input shown in Figure 3.4:

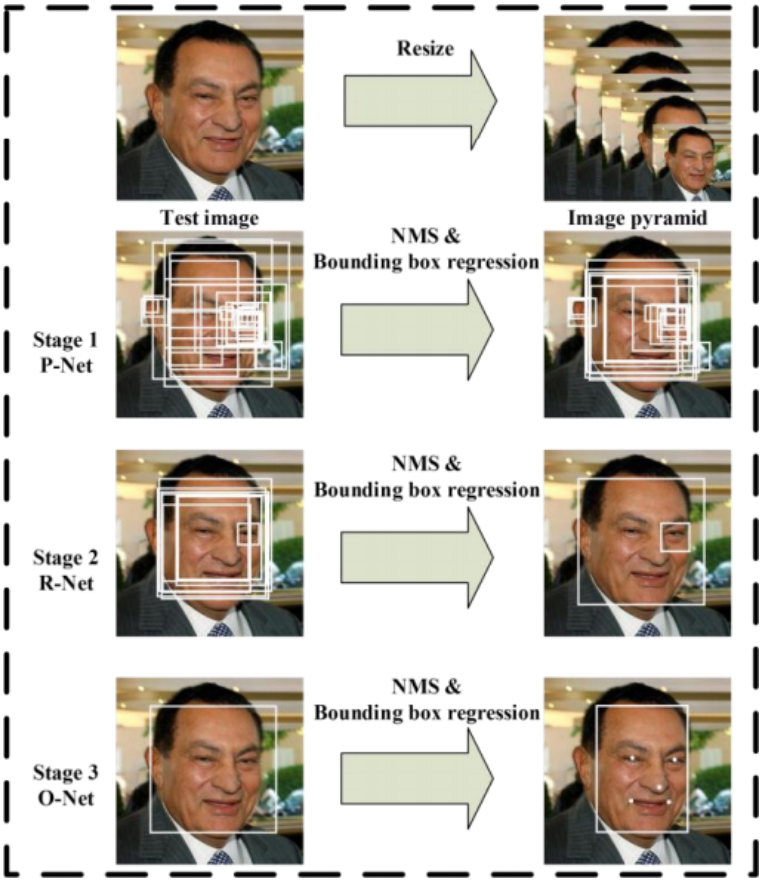

Figure 3.4: (©2016 IEEE) Pipeline of the Three Stages Cascaded Framework [91]

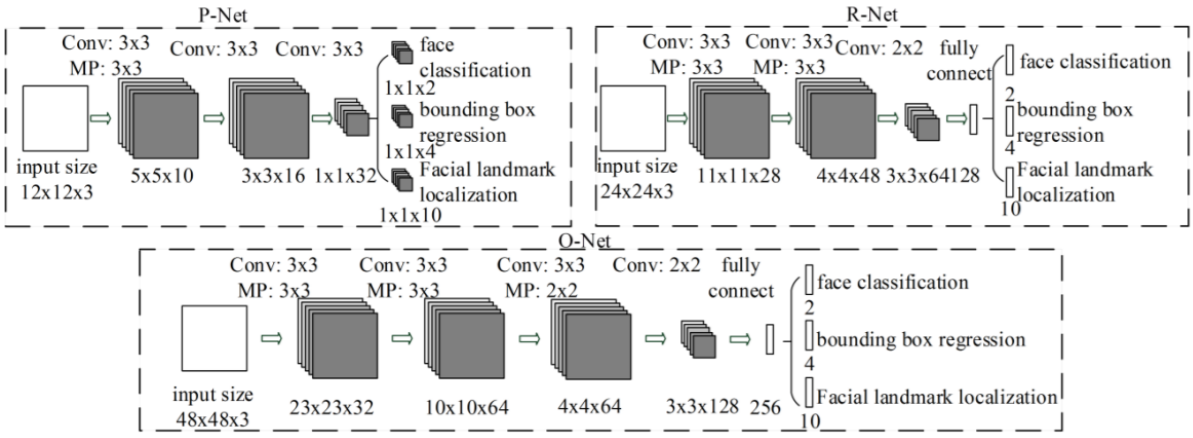

Figure 3.5: (©2016 IEEE) The Architectures of MTCNN [91]

3.1.4 Single Shot Multibox Detector

Single Shot MultiBox Detector (SSD) [51] is a detector that only has one-stage. Unlike other detectors [66] [16] which consist of region proposal generation and bounding box classification. SSD does not re-sample pixels or features inside the bounding box. It detects objects in images using a single feed-forward convolutional network that produces bounding boxes and scores of instances for each object. Then, NMS optimizes the detections [51].

The advantage of a single-stage detector is that it is able to achieve real-time speed because the network generates scores for the presence of each object category in each default box at the prediction time. Afterwards, it produces adjustments to the box of higher score to match the shape of the object. Predictions from multiple feature maps are combined with different resolutions to naturally handle objects of different sizes. SSD eliminates proposal generation and subsequent pixel or feature re-sampling stages in one network. As a result, SSD has shorter runtime compared to two-stages detectors. Then, it is simple to integrate into systems that require fast detection.

Model of SSD

SSD is a one-stage detector, and the main idea is to uniformly assign default boxes densely with different scale ratios in the image. These different scales and aspect ratios

can be used for distribution. CNN can be used to extract features and directly perform classification and regression.

The network produces detections with two features. The first is multi-scale feature maps for detection which adds convolutional feature layers to the end of the truncated base network and allows the predictions of detection at multiple scales and decrease the sizes. The second is convolutional predictors for detection in which case each added feature layer can produce a fixed set of detection predictions using a set of convolutional filters.

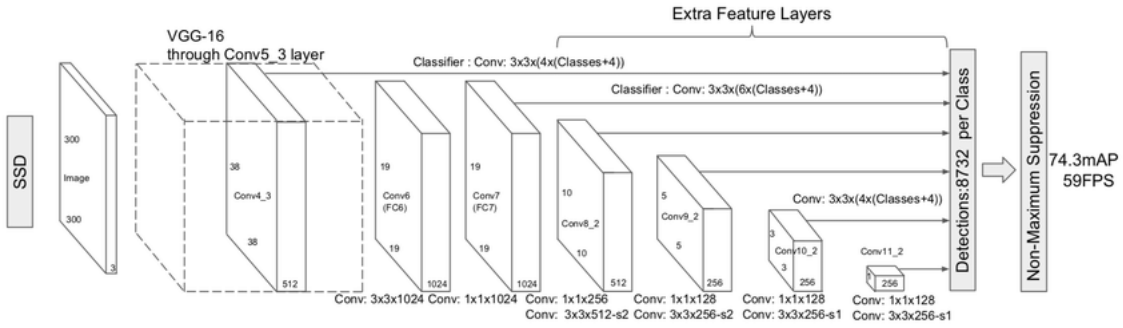

Figure 3.6: The Architectures of SSD [51]

The architecture details of the SSD 300 are shown in Figure 3.6. Convolutional layers $Conv4_3$, $Conv7(FC7)$, $Conv8_2$, $Conv9_2$, $Conv10_2$, and $Conv11_2$ predict both location and confidences. The default box is set with scale 0.1 on $Conv4_3$. For $Conv4_3$, $Conv10_2$ and $Conv11_2$, 4 default boxes are associated with each feature map location. The remaining layers use 6 default boxes. As shown in ParseNet [52], $Conv4_3$ has a different feature scale compared to the other layers. It uses the $l2$ normalization technique introduced in ParseNet [52] to scale the feature norm at each location in the feature map and learn the scale during back propagation.

Matching Strategy of SSD

In SSD, selections of scales and aspect ratios for default boxes are important to determine how it is able to handle different object scales. Pierre et al. [68] and He et al. [26] suggest processing the image at different sizes and combining the results afterwards [51] in

order to produce good bounding box locations of objects. The default boxes correspond to a ground truth detection to train the network. The scale of the default boxes for each feature map is calculated as:

$$s_k = s_{min} + \frac{s_{max} - s_{min}}{m - 1} \, (k - 1) \,, \tag{3.1}$$

where s_{min} is 0.2 and s_{max} is 0.9. These represent the lowest layer scale and the highest layer scale respectively. Every feature map location has 6 default boxes. For each feature map cell, it predicts the offsets relative to the default box shapes in the cell, and it combines all the feature maps. SSD uses different scales and different aspect ratios for the default boxes, as well as the per-class scores to indicate the presence of a class instance in each of those boxes.

Selections bounding boxes from default boxes vary over location, aspect ratio and scale, to match with ground truth boxes. They first match the ground truth box to the default box with the best overlap MultiBox [18]. SSD sets up the ground truth with overlap higher than a threshold of 0.5 IOU to match with the default boxes. An overlap of less than the threshold value of 0.5 IOU is regarded as a negative sample. This not only helps to simplify the learning problem to allow the prediction of high scores but it also ensures only the highest maximum overlapping box is learned. Unlike R-CNN [20]-based models [66] [19], the SSD model does not need to generate regions of interests. As a result, SSD has been used for vehicles detection, airplane detection and other fast-moving objects.

3.1.5 Faster R-CNN

Faster R-CNN is a single unified network for object detection, whose architecture has been improved based on R-CNN [20] and Fast R-CNN [19]. In recent advanced object detection methods, region proposal methods and region-based convolutional neural networks (R-

CNNs) [21] have been used to detect objects for high accuracy requirements. The latest incarnation, Faster R-CNN [66], achieved shorter run time than Fast R-CNN [19].

R-CNN

The R-CNN paper by Girshick et al. [21] was among the first modern incarnations of convolutional network based detectors. The challenge on image classification [40] runs a neural net classifier on cropped and computed box proposals in the image. However, this approach can be very expensive because R-CNN extracts region proposals by selecting a huge number of regions, where the selective search [66] extracts approximately 2,000 regions from the image. First, selective search requires generation of initial sub-segmentation, and obtaining many candidate regions. Second, they recursively combine similar regions into larger ones. Third step is to use the generated regions, produce the final candidate region proposals, and compute the CNN features. Finally, it classifies the regions.

Ren et al. [66] observe the following few problems of R-CNN:

- It takes a substantial amount of time to train the networks since 2,000 region proposals per image need to be classified.
- It takes 47 seconds with the VGG 16 network to process an image on PASCAL VOC 2,007 dataset on average. In comparison to SSD, it is significantly slower.
- A poor candidate region proposal could occur since the selective search algorithm is fixed, resulting in no learning at this stage.

Faster R-CNN structure

A modified algorithm, named "Faster R-CNN" is faster than R-CNN and Fast R-CNN because it feeds the input image to a CNN to generate a convolutional feature map rather than select 2000 region proposals first. Both R-CNN and Faster R-CNN utilize the selective search method to seek the region proposals, which is time consuming because every single

region proposal independently goes through CNN. However, Faster R-CNN uses the Region Proposal Network (RPN) to replace the selective search. The predicted region proposals use Region of Interest pooling layer, which is used to classify the image within the proposed region. Then, Faster R-CNN uses bounding box regression to calibrate the anchors to get the precise proposals. Faster-RCNN uses RPN and shares the Convolutional Neural Networks with the proposal feature maps. Therefore, it reduces the region proposals from 2,000 to 300 [66] [33] to reduce the computational cost.

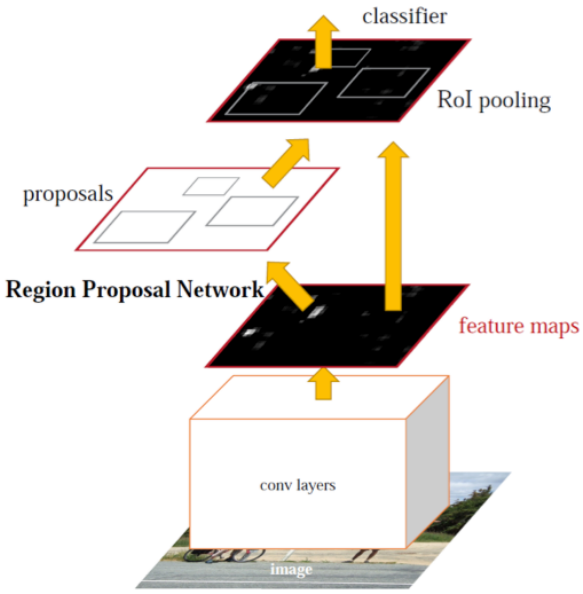

Figure 3.7: Faster R-CNN Structure [66]

Region Proposal Networks (RPN)

The Region Proposal Network replaces the Selective Search method from R-CNN. Figure 3.8 demonstrates the center of the sliding window which has the center points with k number of anchor boxes with different scales and aspect ratios. Ren et al. [66] set the feature map $k = 9$ anchor boxes. It uses softmax to classify the anchors as positive or negative boxes. After, the network applies a proposal layer to obtain accurate proposals by calculating bounding box regression. The lower dimensional feature is fed into the bound-

ing box regression layer because the last proposal layer synthesizes the positive anchors and corresponding bounding box regression offset while excluding small and out-of-bounds proposals. Finally, the proposal layer generates the location of the object bounding box.

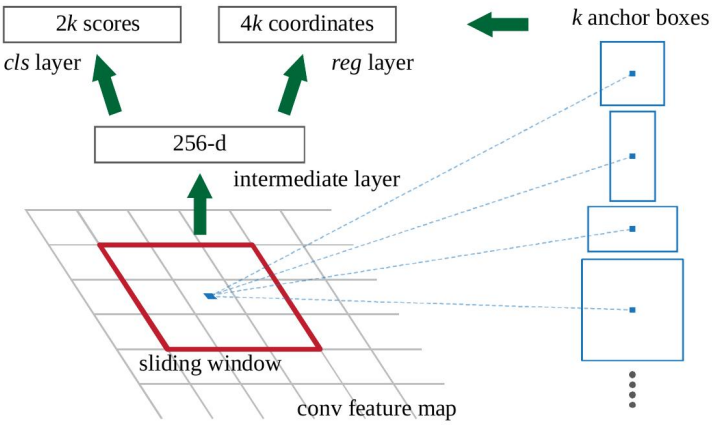

Figure 3.8: Region Proposal Network [66]

After the convolutional feature map, at each sliding window location, it predicts multiple region proposals, and denotes k as the maximum region proposals at each location. The regression layer has $4k$ outputs encoding the coordinates of k boxes because every anchor has (x, y, w, h) 4 values. The classification layer outputs $2k$ scores because every anchor has positive and negative values from the output probability of the object for each proposal.

The RPN loss function consists of the sum of all classification losses and the sum of all regression losses.

$$L(p_i, t_i) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*), \tag{3.2}$$

where p_i is the predicted probability of being an object for anchor i, t_i is the coordinates of the predicted bounding box for anchor i, N_{cls} is the number of anchors in mini-batch (approximately 256), N_{reg} is the number of anchor locations (approximately 2400). p_i^* is

the ground truth of objectness label and t_i^* is the ground truth of objectness bounding box.

3.1.6 R-FCN

Besides Faster R-CNN [66] and SSD [51], Dai et al. [16] propose the Region-based Fully Convolutional Networks (R-FCN) based on a modified Faster R-CNN to balance out the speed of SSD and the accuracy of Faster R-CNN. In contrast to previous region-based detectors such as Faster R-CNN that apply a costly region proposed sub-network hundreds of times, the region-based detector is fully convolutional with all computation shared on the entire image. However, CNNs only focus on feature extraction, and not localization. Consequently, it cannot be used directly for object detection. R-FCN proposes a method of position-sensitive score maps to ensure the sensitivity of object localization and to solve the conflicts between translation-invariance in image classification and translation-variance in object detection [16]. Based on the region proposal networks, it adopts fully convolutional image classifier backbones for object detection. Compared to Faster R-CNN, R-FCN is 2.5-20 times faster and achieves 170ms per image on PASCAL VOC 2007 dataset [16].

Two Stages of R-FCN

After R-CNN [21], the two-stage object strategy [21] [26] [66] consists of region proposal and region classification. Deep neural networks for object detection can be divided into two subnetworks: a shared, fully convolutional sub-network, independent of Region of Interest (ROI) [19]. R-FCN consists of shared, fully convolutional architectures, which is the case of Fully Convolutional Networks(FCN) [55]. It uses a series of specialized convolutional layers as FCN output to form a set of position-sensitive score maps to replace the role for region proposal networks in Faster R-CNN [66]. The location information is encoded into score maps. The position-sensitive RoI pooling layer takes the information from these score maps to give the output of the object position. Figure 3.9 shows position-sensitive mapping: k x k = 3 x 3 position-sensitive score maps in the ROI block, and it shows that

pooling is only applied in one of the k^2 maps.

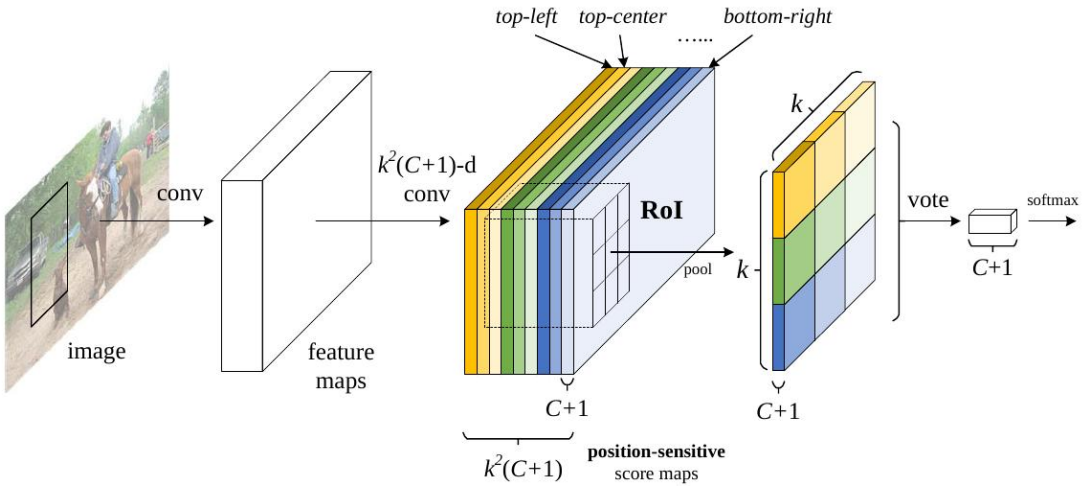

Figure 3.9: RoI in Region based Fully Convolutional Neural Networks [16]

Main Approach of R-FCN

R-FCN is also the two-stage object detection strategy: (1) Region proposals, (2) Region classification. All the learnable convolutional layers are computed on the entire image. Figure 3.10 shows that the last convolutional layer produces a series of k^2 position-sensitive score maps for each class. $k^2(C+1)$ layers represent C classes score maps in addition to one extra class for the background of the image. When $k = 3$, the k x k represent 3 x 3 = 9 position sensitive score maps encoded to an object class. The average pooling is followed by $k^2(C+1)$ layers to obtain C+1 class score maps [16].

3.1.7 Backbone Feature Extractors

From Section 3.1.3 to Section 3.1.6, we present the state-of-the-art detectors with their meta architectures. In this section, three backbones of detectors, also known as Feature Extractors, will be introduced.

Inception V2

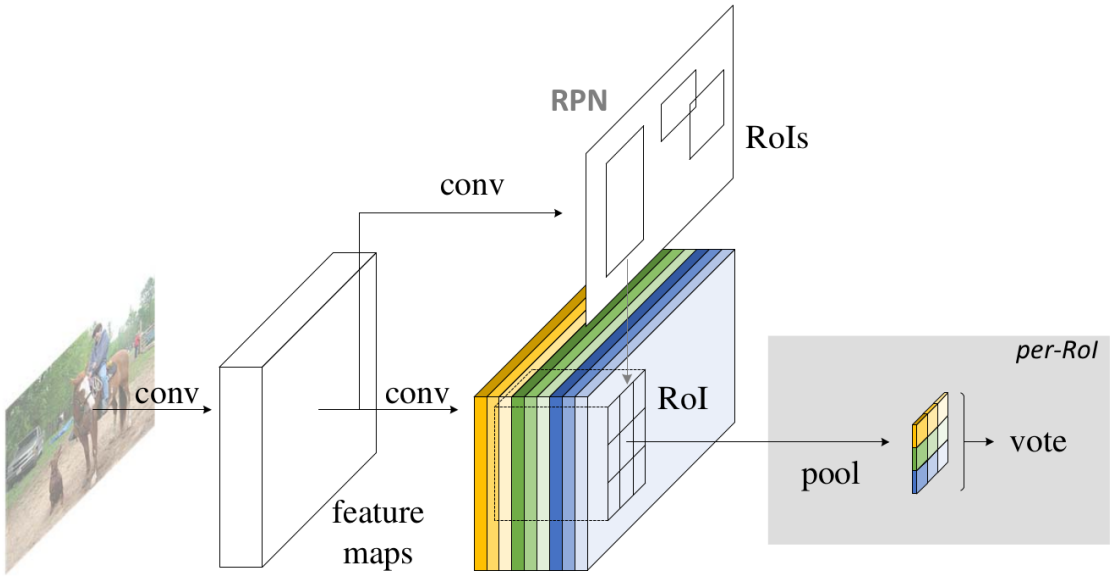

Figure 3.10: Overall architecture of R-FCN [16]

Ioffe et al. [34] set the state-of-the-art performance by using the Inception network in the ILSVRC 2014 class classification and detection challenges. The computational cost of Inception is much lower than VGGNet and its higher performing successors [25]. Szegedy et al. [74] describe the properties of Inception V2 based on large-scale experimentation with various architectural choices of convolutional neural networks. There are four design principles for Inception V2 [74]:

1. Avoid representational bottlenecks with extreme compression.

2. Higher dimensional representations are easier to process.

3. Spatial factorization for lowering computational cost.

4. Balance width and depth of the network to optimize the performance of the network.

There are other ways of factoring convolutions in various settings in order to increase the computational efficiency of the solution. Inception networks are fully convolutional

layers, and reduce the computational cost. Four operations to factorize the convolutional layers exist [74]:

1. Big convolutions can be broken down to smaller convolutions through factorization. For example, a 5 x 5 convolutions can be factorized into two 3 x 3 convolution, sliding this network can be represented by two 3 x 3 convolutional layers which reuses the activation functions between adjacent tiles as shown in the Figure 3.13. Therefore, the computation cost of two 3 x 3 convolutions is (3 x 3 + 3 x 3)/5 x 5 = 18/25 (72%) of the computation cost of a 5 x 5 convolutions.

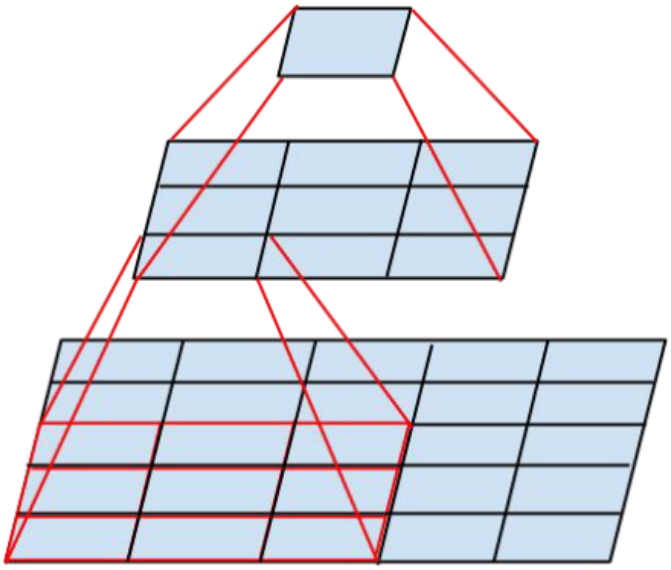

Figure 3.11: (©2016 IEEE) Two 3 x 3 Convolutions Replacing One 5 x 5 Convolutions [74]

2. Spatial factorization can be turned into asymmetric convolutions. This suggests that n x n convolutions can be factorized into a 1 x n and n x 1 convolutions in cascade form. For example, a 1 x 3 convolution followed by a 3 x 1 convolution is equivalent to sliding a two layer network with the same receptive field as in a 3 x 3 convolution. This would give the benefit of 33% fewer calculations for the same number of output

filters by using two layer solution if the number of input and output filters are equal [74].

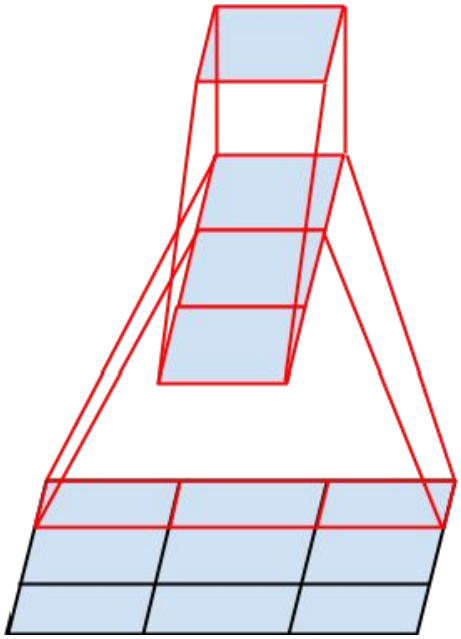

Figure 3.12: (©2016 IEEE) Mini network of 3 x 1 Replacing the 3 x 3 Convolutions [74]

3. Adding a pooling layer and 1 x 1 convolutional layer in parallel reduces the complexity of the networks. Hence, it reduces the computational cost of the bottleneck.

4. Model regularization via label smoothing regularizes the classifier layer by estimating the marginalized effect of label-dropout during training. The marginalized effect of label-dropout occurs when the distribution of the probability of a class $q(k)$ approaches one. Other distributions of the probability for this class approaches zero. The Softmax classifier and its loss function:

$$p\left(k|x\right) = \frac{exp(z_k)}{\sum_{i=1}^{k} exp(z_i)}, \tag{3.3}$$

$$loss = -\sum_{k=1}^{K} log(p(k))q(k). \tag{3.4}$$

Inception network version 2 structure is shown in Figure 3.13.

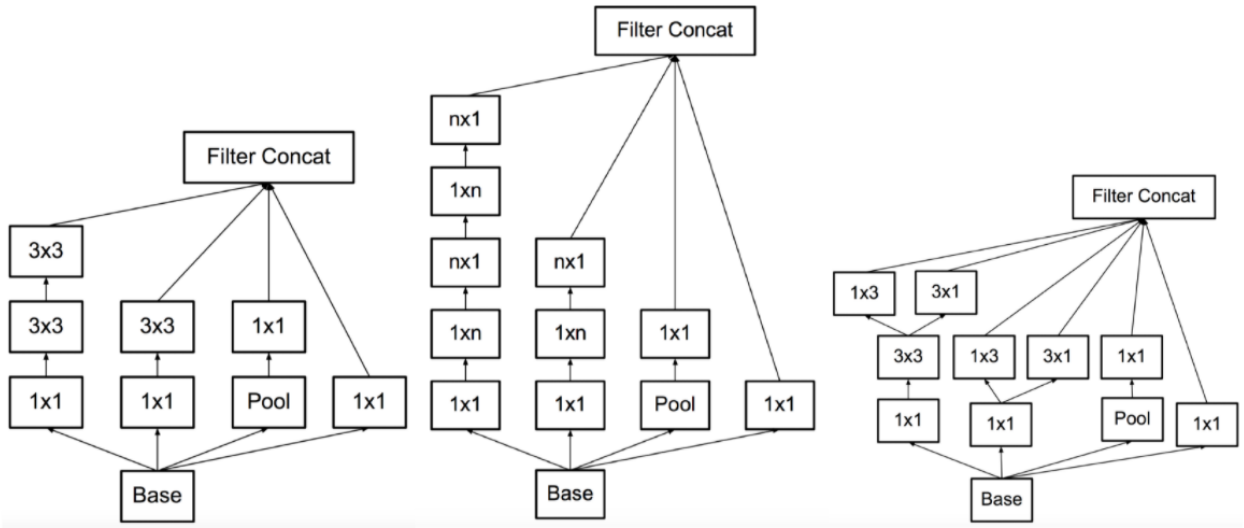

Figure 3.13: (©2016 IEEE) Stage One of Three of Inception [74]

ResNet

Residual Learning Network (ResNet) was first introduced by He et al. [27]. The residual nets with a depth of up to 152 layers are 8 times deeper than VGG nets [71] and still maintain a lower run time. Training the deeper neural networks is prone to have degradation problems such as gradients vanishing or exploding. For example, if the initial gradient is slightly less than 1, taking many derivatives through deep neural networks results in gradient vanishing. Likewise, if the initial gradient is slightly bigger than 1, after taking many derivatives, causes the gradient to explode approximately to infinite.

ResNet block [27], denotes the desired underlying mapping as H(x). As demonstrated in figure 3.14, the formulation of $F(x) + x$ acts as a feed-forward neural network [22] with "shortcut connections" that are skipping one or more layers. The shortcut connections simply work as identity mapping, and their outputs are added to the outputs of the stacked layers. Therefore, every time the shortcut pulls the result of gradient to 1 through this identity mapping to ensure that every gradient result does not go towards infinity or 0

after taking many derivatives through deep neural networks.

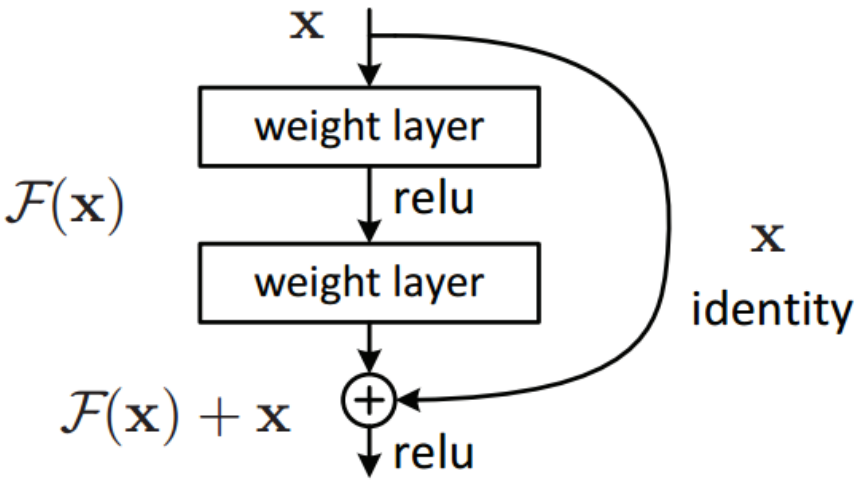

Figure 3.14: (©2016 IEEE) The Building Block of Residual Learning [27]

The residual is the other path that has the function $F(x) := H(x) - x$, and the feed-forward network $F(x) = x$, learning the residual of input and output of two previous functions, which is more efficient because it simplifies the network (only aims to learning the residual network rather than the whole network). Our implementation uses ResNet101 consisting of 101 total layers as the backbone structure for a multi-face detector [27]. The table of ResNet with different layers is shown in Figure 3.15:

Mobilenet V1

The MobileNet [32] model is based on depth-wise separable convolutions. This is a fast processing model for mobile and embedded vision applications. The MobileNet network describes two model shrinking hyper-parameters width multiplier and resolution multiplier. The width multiplier brings the network down to small segments along the width of the network. The resolution multiplier changes the resolution of the input images to lower the representation in each layer.

This model is a form of factorized convolutions to factorize a standard convolution into a depth-wise convolution and a 1 x 1 convolution called a point-wise convolution. In terms

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 3.15: (©2016 IEEE) Table of Different Architectures of ResNet [27]

of deep convolutional neural networks, Figure 3.16 shows a D_K x D_K with length of N and width of M convolutional filter. This standard convolutional filters can be split in two parts. Figure 3.17 shows point-wise convolutional filters, D_K x D_K with the depth of 1. Figure 3.18 shows depth-wise convolutional filters, 1 x 1 convolutional filters with the depth of M (point-wise convolution in the context of depth-wise separable convolution).

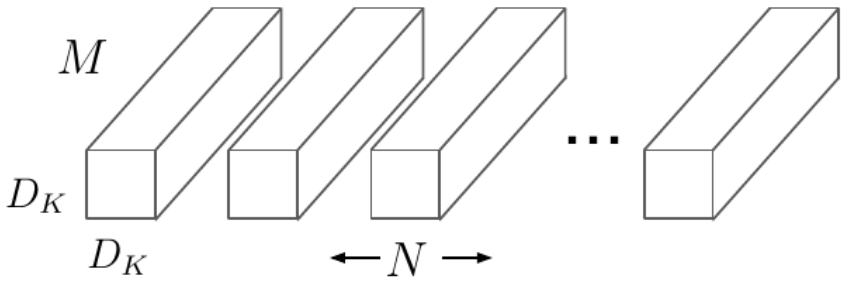

Figure 3.16: Standard Convolutional Filters [32]

MobileNet constructs smaller and less computationally expensive models using width multipliers transforming to a thin network uniformly at each layer. Applying resolution multiplier reduces the resolution to lower the computational cost of a neural network. Since this network is highly efficient, this can be easily implemented in cellphones or other small

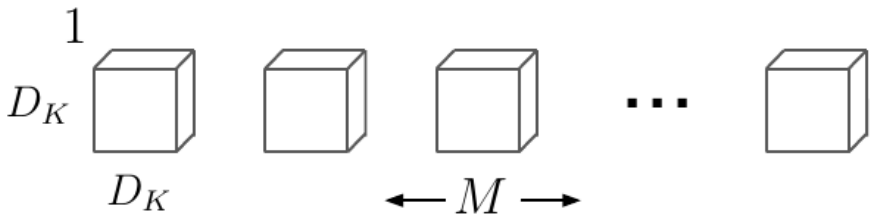

Figure 3.17: Depth-wise Convolutional Filters [32]

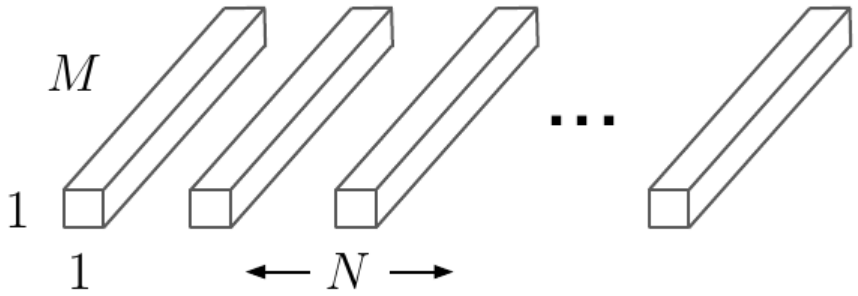

Figure 3.18: 1 x 1 Point-wise Convolutional Filters [32]

devices, so it is named MobileNet.

3.2 Multi-Face Tracking Systems

For safety and security development, many cameras are installed in urban areas. Multi-face tracking analysis in videos is becoming increasingly important, and multi-face tracking has been a popular subject in academia and in industries. Yang et al. [86] reviewed advanced visual tracking methods which can be used for face tracking and listed a variety of common tracking algorithms such as tracking-by-detection, adaptive discriminative generative model, etc. Chrysos et al. [15] evaluated face tracking by applying deformable tracking through hybrid approaches and by using face detection, model free tracking and facial landmark localization technologies. However, previous methods mentioned in Yang et al. [86] are less successful in tracking and re-identification. Meanwhile, while the proposed method from Chrysos et al. [15] could not achieve real-time requirements. In this section, we introduce real-time Multi-Face Tracking Systems which exclusively focus on multi-object tracking (MOT) problems for faces based on a previous MOT tracking algorithm [83]. Bewley et al. [7] explored a pragmatic approach associating objects efficiently for online and real-time MOT. The MOT problem can be viewed as a data association problem where the objective is to associate detection across frames in a video sequence. Trackers associate detection and prediction data by using various methods for linking the motion [17][89] and appearance [8] [39] of objects in the scene. Traditionally, tracking-by-detection with data association is implemented by Joint Probabilistic Data Association Filters which are statistical methods to problems of object-measurement assignments in tracking algorithms such as using a Kalman Filter [28] [14]. Alternatively, the Hungarian Algorithm can be used to optimize the detection and predicted motion between current frame and next frame. Traditional data association connects physical motions. Features of objects aid to associate objects across frames. In this section, we will focus our discussion

on a multi-face tracking system based on deep association matching cascade [83] with four different classifier loss functions.

3.2.1 Data Association

Data Association for tracking is the association between current detection data and candidates. A simple strategy is to gate the motion window around the prediction. To understand Data Association, let us consider a different tracking "paradigm". First, detect objects in each frame and figure out inter-frame correspondence between them, such as two-frame matching (current frame to predicted frame matching). The popular techniques are Nearest Neighbor Data Association (NNDA) [2], Probabilistic Data Association (PDA) [38] [14] and Joint Probabilistic Data Association (JPDA) [24] [75] [28].

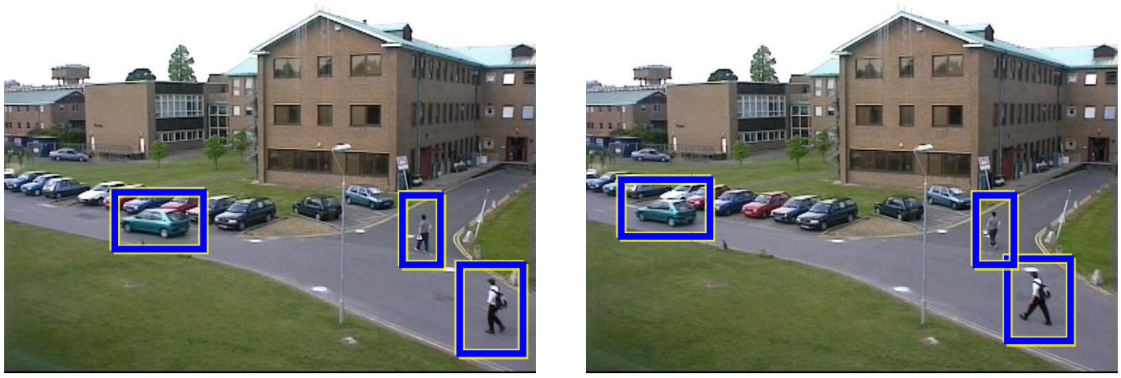

Figure 3.19: (©2015 IEEE) Two-frame Matching(Correspondence Problem) [24]

Multi-frame matching observes a set of tracked trajectories in a new frame. We want to recursively estimate the current state every time that a measurement is received. The first step consists of prediction by propagating state Probability Density Function (PDF) forward in time, which takes the process noise such as unstable motion, occlusion into account. The second step is the gate. Gating is a method for pruning matches that are geometrically unlikely from the start, by setting a threshold for determining possible matching observations. The third step determines the best match based on estimators.

The last step is to modify the prediction PDF based on current measurement for the update data. At this stage, we obtain a matrix with all the unmatched predictions and detections, and apply the assignment cost matrix which computes as the intersection-over-union (IOU) distance between each detection and all predicted bounding boxes from the existing targets.

Kalman Filter

Kalman Filter is an algorithm used to calculate the future state by taking a series of measurements observed over time with noise and uses the covariance and mean values from the past and the current system state to predict the object positions.

In video face tracking, each track k for the face counts the number of frames since the last successful measurement association a_k. This counter is incremented during the Kalman Filter. The estimate is updated using a weighted average with the outcome of the next measurement and more weight is given to estimates with higher certainty. It recursively keeps updating the present input measurement based on the calculation of the previous state with the uncertainty matrix.

The Kalman filter process mainly is divided into two steps:

- The prediction step uses a previous estimated state and the linear model to predict the value of the next state with the state estimate covariance:

$$\hat{X}_{k|k-1} = F\hat{X}_{k-1|k-1} + Bu_k \tag{3.5}$$

$$P_{k|k-1} = FP_{k-1|k-1}F^T + Q \tag{3.6}$$

where $\hat{x}_{k|k-1}$ is a posteriori state estimate at time k; F is the state-transition model which is applied to the previous state x_{k-1}; B is the control-input model which is applied to the control vector u_k; P is the posteriori error covariance matrix which is

a measure of the estimated accuracy of the state estimate; and Q is the covariance of the process noise.

- The update step uses the current measurement of the output together with the statistical properties of the model in Equation 3.6, to correct the state estimate. The values calculated are the innovation covariance in Equation 3.10, the Kalman gain in Equation 3.8 resulting in the updated state estimate and state estimate covariance:

$$S_k = HP_{k|k-1}H^T + R \tag{3.7}$$

$$K_k = P_{k|k-1}H^T S_k^{-1} \tag{3.8}$$

$$\hat{x}_{k|k} = F\hat{x}_{k|k-1} + K_k(Z_k - H\hat{x}_{k|k-1}) \tag{3.9}$$

$$P_{k|k-1} = (I - K_k H)P_{k|k-1} \tag{3.10}$$

Where S is the innovation covariance; H is the observation model; R is the covariance of the observation noise; K_k is the optimal Kalman gain; the Kalman filter model assumes the true state at time k derives from the state at $(k-1)$ according to $\hat{x}_k$; Z_k is the observation according to the true state $\hat{x}_k$. These two steps are repeated for every sample: $k = 1, 2, \ldots, K$.

Hungarian Algorithm

The Hungarian Algorithm is an algorithm used to solve the assignment problem. The general idea is to find the best assignment by sorting the distance matrix in tracking. Hence, applying this algorithm to multi-face tracking provides efficient matching of frame-to-frame faces.

The Hungarian Algorithm procedures are:

1. Row Reduction: Subtract the smallest score in each row.

2. Column Reduction: Subtract the smallest score in each column.

3. Draw as few row and column lines as possible to cover all the zeros, if the number of lines drawn are less than the number of rows or columns, go to step 4. Otherwise, go to step 6.

4. If the number of lines drawn is less than the number of rows or columns, modify the table by following the steps below:

   - Subtract the smallest uncovered number from every uncovered number in the table.
   - Add the smallest uncovered number to the numbers of intersections of covering lines.
   - Numbers crossed out but at intersections of cross-out lines carry over unchanged to the next table.

5. Repeat steps 3 and 4 until every row and column have zeros.

6. Make the assignment by beginning with rows or columns with only the zeros that these rows and columns have. Cross out both the rows and the columns after the match.

The example below illustrates that we have 5 objects in the current frame and 5 objects in the predicted frame, and that we can build a table of match scores $m(5, 5)$ to find the minimum matching scores.

$$\begin{array}{ccccc} 0.23 & 0.17 & 0.62 & 0.33 & 0.05 \\ 0.18 & 0.59 & 0.19 & 0.25 & 0.98 \\ 0.08 & 0.66 & 0.99 & 0.86 & 0.51 \\ 0.11 & 0.51 & 0.82 & 0.39 & 0.32 \\ 0.65 & 0.86 & 0.11 & 0.82 & 0.21 \end{array}$$

For the cost calculation, the first step is to subtract the minimal cost (number in this case) from each row. This ensures that every row has at least one 0.

0.18	0.12	0.57	0.28	0
0	0.41	0.01	0.07	0.80
0	0.58	0.91	0.78	0.42
0	0.40	0.71	0.28	0.21
0.54	0.75	0	0.71	0.10

The second step is to subtract the minimal cost (number in this case) from each column, and ensure that every column has at least one 0.

0.18	0	0.57	0.21	0
0	0.29	0.01	0	0.80
0	0.46	0.91	0.71	0.42
0	0.28	0.71	0.21	0.21
0.54	0.63	0	0.64	0.10

The third step is to strike through the rows and columns that have zeros. We observe that four lines are drawn to cover all zeros, and four lines are less than the number of rows or columns which are five lines. Then, we proceed the fourth step.

~~0.18~~	~~0~~	~~0.57~~	~~0.21~~	~~0~~
~~0~~	~~0.29~~	~~0.01~~	~~0~~	~~0.80~~
0	0.46	0.91	0.71	0.42
0	0.28	0.71	0.21	0.21
~~0.54~~	~~0.63~~	~~0~~	~~0.64~~	~~0.10~~

The fourth step from the elements that are left, find the lowest value not being struck which is 0.21 in the table. Subtract this from all elements that are not struck. Then, add the lowest value to elements that are present at the intersection of two lines.

0.39	Ⓞ	0.57	0.21	0
0.21	0.29	0.01	Ⓞ	0.80
Ⓞ	0.25	0.70	0.50	0.21
0	0.7	0.50	0	Ⓞ
0.75	0.63	Ⓞ	0.64	0.10

Now we form a permutation matrix with the zero elements in every row and column, and this forms the matching with minimum cost.

The last step is to check if the matching is possibly done. If not, we execute the third step until we form a minimization assignment. The zeros in the circle present the best matching from the original table given in each row and column.

0.23	0.17	0.62	0.33	0.05
0.18	0.59	0.19	0.25	0.98
0.08	0.66	0.99	0.86	0.51
0.11	0.51	0.82	0.39	0.32
0.65	0.86	0.11	0.82	0.21

3.2.2 Simple Online and Real-time Tracking

Simple Online and Real-time Tracking (SORT) [7] associates objects from their motions for online and real-time multi-object tracking. It is a tracker that has low runtime complexity . It uses a Kalman filter in image space and performs frame-by-frame data association techniques with the Hungarian Algorithm based on an association metric that measures bounding box overlap. SORT does not handle occlusion or objects re-entering scenes. SORT is introduced as a baseline method for developing a modified algorithm [82] by Wojke et al. that is discussed in Section 3.2.3.

Estimation Model

The object model including its representation and the motion model are used to propagate a target's identity into next frame. The model equation is:

$$x = [u, v, s, r, \dot{u}, \dot{v}, \dot{s}]^T , \tag{3.11}$$

where u and v represent the horizontal and vertical pixel location of the center of the target in the video frame, scale s and r represent the scale (area) and the aspect ratio of the target's bounding box respectively [7]. The aspect ratio r is noted to be a constant. $\dot{u}$ is the vector differential horizontal pixel location with respect to time, and $\dot{v}$ is the first derivative of the vector in vertical pixel location with respect to time; likewise $\dot{s}$ is first derivative of the scale area with respect to time. Intuitively the first four variables represent the current state and the last three variables represent the update state.

When a detection is associated with a target, the detected bounding box is used to update the target state where the velocity components are solved optimally via a Kalman filter framework [4]. If no detection is related to the target, the state is simply predicted without correction using the linear velocity model. The IOU distance of each detection can be applied. The IOU threshold is set to be 0.3 to reject the detection with 0.3 IOU overlap. All predicted bounding boxes calculate the assignment cost matrix for matching with detection bounding boxes.

Track Identities

If the overlap between detection bounding box and target bounding box is less than the IOU threshold, the track is regarded as being deleted. Tracker identities are defined as unique identities of objects entering and leaving the scene, and they need to be created or deleted, respectively. When there is no matching between the detection and target within frames T_{Lost} , the track is deleted. However, Bewley et al. [7] set T_{Lost} to be 1 because the constant velocity model is a poor predictor of the true dynamics. Therefore, it becomes a simple detection and tracking within only 1 frame for associating the prediction

and observation. Moreover, the fast deletion of lost targets improves the efficiency of this tracker. Overall, SORT aims at frame-to-frame associations to grow tracklets which has the lowest number of lost targets in comparison to the other methods such TDAM [48] and MDP [85].

3.2.3 Deep Association Metric for SORT

The SORT algorithm returns a relatively high number of Identity Switches(IDS). On the MOT challenge dataset [41], SORT with a state-of-the-art pedestrian detector [66] ranks on average higher than other trackers [65] [39] on standard detections. However, it cannot maintain good tracking accuracy when the object is covered by another object for a long period of time. As a result, high number of IDS reduces the SORT association metric accuracy for a short period of time. Therefore, an improved version of SORT with a deep association metric, named Deep SORT [83], helps to solve long-term occlusion and further decrease the tracking loss. Deep SORT combines both physical motion and feature appearance information to overcome the occlusion problem, and it decreases the rate of IDS at in the end.

Tracking and Assignment problem

The track handling and state estimation are similar to simple SORT [7]. However, the tracking scenario is defined on the eight dimensional state space to describe the motion status:

$$x = \left(u, v, r, h, \dot{x}, \dot{y}, \dot{r}, \dot{h}\right), \tag{3.12}$$

where the bounding coordinates (u, v, r, h) are taken as direct observations of the object motion status. As indicated in the discussion of SORT previously, (u, v) is the central coordinate of the bounding box, r is the aspect ratio, and h is the height, rest of the parameters $\dot{x}, \dot{y}, \dot{r}, \dot{h}$ are the speed information in the frame-to-frame of the image coordinate based on the constant speed model and linear observation.

There is a threshold for tracking through recording the time from the last successful match to the current state. When the value is greater than the threshold maximum age A_{max} set in advance, the object tracking with its pre-defiend ID is considered to be terminated. Intuitively, the tracking that has not been matched for a period of time is deleted. A new enumeration for the single target track state can be defined by the following three types of track state: (1) Tentative, (2) Confirmed, and (3) Deleted. Newly created tracks are labeled as "Tentative" and those tracks for detections may be false alarms until sufficient matched physical motion and feature appearance are collected. The track state is changed to "Confirmed". The "Deleted" candidate is removed from the set of active tracks after maximum age A_{max} [83].

The squared Mahalanobis distance computes the metrics of each object's physical motion distance based on the standard deviations away P from the mean value of D. The squared Mahalanobis distance calculates the predicted Kalman states and newly arrived measurements:

$$d^{(1)}(i,j) = (d_j - y_i)^T S_i^{-1} (d_j - y_i), \tag{3.13}$$

Equation 3.13 represents the matching between the i-th tracking trajectory in the measurement space by (y_i, S_i). The j-th detection by d_j. $(d_j - y_i)$ presents the multi-dimensional distances from mean values, and S_i^{-1} represents the inverse of covariance matrix. Taking this into consideration with regards to continuous association, a decision equation can be formed by setting the confidence interval of Mahalanobis distance at 95% computed from the inverse x^2 distribution. The decision equation is defined by:

$$b_{i,j}^{(1)} = ¶\left[d^{(1)}(i,j) \leq t^{(1)}\right]. \tag{3.14}$$

The threshold is set to $t^{(1)} = 9.4877$. The Mahalanobis distance performs well without

occlusion during tracking because it emphasizes physical motion in the trajectory. When tracking through occlusions over a long period of time, a second metric based on feature appearance distance is integrated into the assignment problem. This second metric measures the smallest feature distance between the i-th track and j-th detection in feature appearance space:

$$d^{(2)}(i,j) = min\{1 - r_j^T r_k^{(i)} | r_{k^i} \in R_i\}, \tag{3.15}$$

where the metric r_j is set to $|r_j| = 1$, the gallery $R_k = \{r_k^{(i)}\}_{k=1}^{L_k}$ stores the last 100 associated feature appearance in deep feature distance for each track k.

As the decision equation for physical motion distance above, the feature distance for feature appearance space also has the decision equation defined as:

$$b_{i,j}^{(2)} = ¶\left[d^{(2)}(i,j) \leq t^{(2)}\right]. \tag{3.16}$$

There are two benefits of this method. (1) The Mahalanobis distance for short-term predictions provides information about the possible face locations based on physical motions (2) The feature distance gives information of appearance which is used to recover identities after long-term occlusions. The final cost takes the combined weighted sum of both metrics the association problem.

$$c_{i,j} = \lambda d^{(1)}(i,j) + (1-\lambda) d^{(2)}(i,j), \tag{3.17}$$

where λ is weight for controlling the combined association cost. Finally, the tracking assignment obtains a successful association which is admissible within the gating region of both metrics:

$$b_{i,j} = \prod_{m=1}^{2} b_{i,j}^{(m)}. \tag{3.18}$$

3.2.4 Classifier Loss Functions

The Deep SORT system embeds a CNN for feature extraction to help solve occlusion of faces. We present different classifiers in this tracking algorithm to distinguish different faces, and review four classifier loss functions: Cosine Softmax Classifier [82], Angular Softmax Loss [53], Magnet Loss [67], Triplet Loss [29].

Cosine Softmax Classifier

Recall that a standard Multi-Class Softmax Classifier:

$$p(y = k \mid r) = \frac{exp({w_k}^T r + b_k)}{\sum_{n=1}^{C} exp({w_n}^T r + b_n)}, \tag{3.19}$$

where $r = f(x), r \in \mathbb{R}$ is the feature representation. For example, a training set has N images within a number of C classes. The numerator in Equation 3.19 stands for the scores of the class k in the image computed with the weight and bias factors. The denominator represents the total scores of all C classes. The binary classification is presented when $C = 2$. Finally, the standard Softmax function classifies the highest probability scores among C class scores.

However, the standard Softmax Classifier can be modified with a few changes [82]. First, the bias terms b_K has been removed for all classes, it reduces all the biases parameters to null compared to the standard softmax classifier. Second, adding the free scaling parameter κ to be the marginal factor for each class. Additionally, the weights are normalized to unit length, i.e., $\tilde{w_k} = w_k / \parallel w_k \parallel_2, \forall_x = 1, ..., C$ [82]. Therefore, the modified softmax classifier

forms a cosine formation, and this cosine softmax classifier is represented by

$$p(y = k \mid r) = \frac{exp(\kappa \cdot \tilde{w_k}^T r)}{\sum_{n=1}^{C} exp(\kappa \cdot \tilde{w_n}^T r)}. \tag{3.20}$$

The lower κ creates less discriminative margin and the higher κ places larger penalty on misclassified samples to give higher discrimination.

Angular Softmax Loss

The standard softmax classifier for multi-class classification in Equation 3.19 above can also be modified to another softmax classifier loss function. First, weights and bias terms $w_i^T x + b_i$ can be rewritten as $\parallel {w_i}^T \parallel \parallel x \parallel cos(\theta_i) + b_i$ such that Equation 3.20 changes to:

$$L_i = -\log(\frac{exp(\parallel w_{y_i} \parallel \parallel x_i \parallel \cos(\theta_{yi}, i) + b_{yi})}{\sum_j exp(\parallel w_i \parallel \parallel x_i \parallel \cos(\theta_{j,i}) + b_j)}), \tag{3.21}$$

where $\theta_{ji} (0 \leq \theta_{j,i} \leq \pi)$ is the angle between vectors w_j and x_i. Similar to cosine softmax loss, the weight can be normalized to $\parallel w_j \parallel = 1$ and the bias term can be removed, then we have the modified softmax loss

$$L_{modified} = \frac{1}{N} \sum_i -\log(\frac{exp(\parallel x_i \parallel \cos(\theta_{yi}, i)}{\sum_j exp(\parallel x_i \parallel \cos(\theta_{j,i}))}). \tag{3.22}$$

By adding the marginal term $m (m > 2)$ into the Equation 3.22, there is more constrained classification. The finalized version of angular softmax loss is

$$L_{ang} = \frac{1}{N} \sum -\log(\frac{exp(\parallel x_i \parallel \cos(m\theta_{yi}, i))}{exp(\parallel x_i \parallel \cos(m\theta_{yi}, i)) + \sum_{j \neq y_i} exp(\parallel x_i \parallel \cos(\theta_{j,i}))}). \tag{3.23}$$

When we increase variable m, the angular margin increases. This classification between features from different classes creates strong discriminative ability to classify feature

identities via the angular margin.

Magnet Loss

Magnet loss [67] divides or groups its representation space by identifying intra-class variations and inter-class similarity. In the K cluster assignments A_1^c,...,A_k^c with K classes, the minimum intra-cluster distance can be written as

$$A_1^c, ..., A_k^c = arg \min_{A_1^c,...,A_k^c} \sum_{k=1}^{K} \sum_{r \in A_k^c} \parallel r - u_k^c \parallel_2^2 . \tag{3.24}$$

The cluster center $r - u_k^c$ in N classes with class representation r can be represented by

$$r - u_k^c = \frac{1}{N} \sum_{r \in A_k^c} r. \tag{3.25}$$

For multi-face re-identification task, the loss can be written as

$$L_m(y, r) = \{- \log(\frac{exp(-\frac{1}{2\sigma^2} \parallel r_n - u(r_n) \parallel_2^2)}{\sum_{k \in C(r_n)} exp(-\frac{1}{2\sigma^2} \parallel r_n - u_k^c \parallel_2^2)})\}_+, \tag{3.26}$$

similar to Equation 3.24, where r_n is the representation of the class being clustered, $u(r_n)$ is the center of the clustering, $C(r_n) = \{1, ..., C\}$ as the class of representation from the multi-class n. σ^2 is the variance of all samples away from their class mean, and $\{.\}_+$ is the hinge function. From this equation, $\parallel r - u(r_n) \parallel_2^2$ increases with the distance between their inter-class, and $\parallel r_n - u_k \parallel_2^2$ decreases with the distance for intra-class variation. It demonstrates good performance for shortening inter-class similarities and pushing away intra-class variations.

Triplet Loss

Triplet loss [81] is defined over tuples of points given the triplet relation [29] [82]

$$L_{tri}(r_a, r_p, r_n) = \{m + \parallel r_a - r_n \parallel_2 - \parallel r_a - r_p \parallel\}_+, \tag{3.27}$$

Name	Patch Size/Stride	Output Size
Conv 1	3 x 3/1	32 x 128 x 64
Conv 1	3 x 3/1	32 x 128 x 64
Max Pool 3	3 x 3/2	32 x 64 x 32
Residual 4	3 x 3/1	32 x 64 x 32
Residual 5	3 x 3/1	32 x 64 x 32
Residual 6	3 x 3/2	64 x 32 x 16
Residual 7	3 x 3/1	64 x 32 x 16
Residual 8	3 x 3/2	128 x 16 x 8
Residual 9	3 x 3/1	128 x 16 x 8
Dense 10		128
Batch and l2 normalization		128

Table 3.1: The CNN Architecture for Deep Appearance [82]

where r_a is the anchor point, r_p is the positive point, r_n is the negative point, and the $\{.\}_+$ denotes the hinge loss function for maximum-margin classification. The distance difference between the negative and positive pair is larger than a predefined margin m. Hence, the loss ensures that when given an anchor point, the projection of a positive point belonging to the same object is closer to the anchor's projection than that of a negative point belonging to another object [29].

3.2.5 Deep Feature Appearance

To ensure deep features succeed in tracking with a matching cascade, a well-discriminating feature needs to be trained and embedded into matching cascade feature classifiers. Wojke et al. [82] use a 15 layers CNN, including two convolutional layers in each residual block. The CNN architecture contains two convolutional layers, pass to a max-pooling layer followed by 6 residual blocks. The global feature map of size 128, as shown in Table 3.1, illustrates input to the dense layer 10 followed with the final batch and l2 normalization. The output features can be trained with the four different feature classifier loss functions from section 3.2.4.

3.3 Summary

In this chapter, we reviewed a dedicated multi-face detector, three common object detectors and three feature extractors. We also presented a deep feature association metrics for multi-object tracking and four feature classifier loss functions

In the next chapter, we will present the proposed methods for multi-face detection and tracking system based on the detectors from this chapter and deep feature association metrics with different feature classifiers.

Chapter 4

Proposed Approach

A state-of-the-art multi-face tracking system can be built using tracking by detection:

- Deep face detectors can be trained using standard object detector architectures
- Face associations can be found with deep appearance metrics in a matching strategy

Since Deep SORT was never performed in multi-face tracking, we adapt a multi-face detection and tracking system based on evaluation on faces and common object detections trained on faces in addition to a multi-face adaptation matching cascade from Deep SORT.

An overview the entire multi-face detection and tracking system is provided in the flow diagram (the red highlights are our modifications and add-ons) in Fig. 4.1. In this chapter, we discuss details of all the steps.

4.1 Dataset and Benchmark

Datasets are used to train, test or evaluate a machine learning model. In our approach, we use different datasets which consist of either a dozen of videos or over thousands of images with annotated faces. Videos which all have over thousands frames are extracted to image

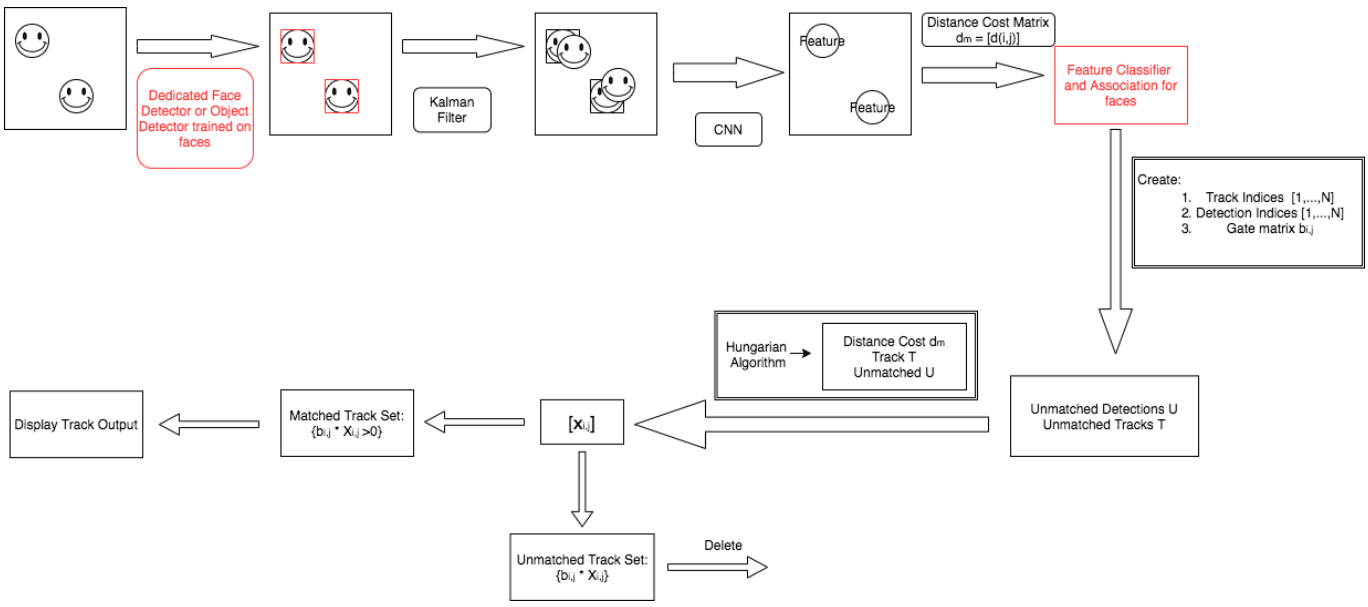

Figure 4.1: Flow Diagram of Multi-face Detection and Tracking System

frames. Each dataset is divided into three parts: (1) Training Set, (2) Validation Set, (3) Testing Set.

- The training set aims to optimize the networks' weights and biases through back-propagation. It also builds the model for "self-learning" objects.
- The valuation set is to evaluate how the model performs. The fixed and limited amount of the training set with potentially not enough diversity may result in over-fitting. Cross-validation can reduce overfitting.
- The testing set is to test the model by providing an unbiased dataset for evaluation of the final model.
- The ground truth is the annotated images for supervised learning in order to calculate the accuracy and loss in each model.

A benchmark is a challenge dataset that evaluates a model's performance. In the detection phase, we use WIDER FACE [87], and FDDB [35] datasets for face detector

training, CelebA [54] for face landmarks training in MTCNN detector. In the tracking phase, we use Youtube Faces [84] to train the four classifiers. We perform our adapted Deep SORT for faces on Music Video Benchmark [93]. We notice that the Music Videos dataset has defects on its ground truth for evaluation of faces by using tracking by detection methods because four out of eight clips have faces of audience, pedestrians and partner dance presented in frames, but these faces are not annotated. Conversely, our benchmark dataset includes every face presented in scenes with annotations, and we label it with corresponding ID. Also, they lack multi-face tracking open source dataset, we are giving this convenience to academia. Therefore, we introduce a new benchmark from the Government of Canada – House of Commons, and evaluate trackers on this new dataset.

1. WIDER FACE [87] is a multi-face detection dataset that contains 32,203 images with 39,3703 faces at different scales and diverse environments.

2. CelebA [54] is a face attributes dataset that consists of 10,177 persons with 20,2599 face images. Every face has 5 landmark locations.

3. FDDB [35] is also a dataset for face detection in unconstrained environments. It has annotations of locations of all faces in these images from Sung et al. [73]. FDDB contains 2,845 images with a total of 5,171 faces in both grayscale and color images.

4. The Youtube Faces dataset [84] has 3,425 videos of 1,595 different people. Each person's clips vary from 1 to 6. The shortest video clip has 48 frames, and the longest video clip has 6,070 frames.

5. The Music Video Dataset [93] is an unconstrained multi-face tracking benchmark that consists of 8 music videos from YouTube channels. Three of the videos are live concerts and the other five videos are music television. This benchmark is for multi-face tracking with the challenges of complex background, frequent scenes switches, fast motion of multiple cameras.

6. The House of Commons Dataset. We noticed a lack of multi-face tracking benchmark in unconstrained videos of meetings for open source use in academia. We made the House of Commons video dataset with 7 unconstrained video clips within 183 different people faces for multi-face tracking from the Government of Canada – The House of Commons. These clips have the attributes of face motion blurring, face occlusion, small and low resolution of faces which bring challenges in detection and tracking.

4.2 Multi-face Detection

First of all, a dedicated multi-face detector – MTCNN will be re-trained on WIDER FACE and Celeb A to obtain the frozen detector model, while Celeb A is used for training the landmark of faces and is not used to help detection of faces. The comparison of our multi-face detection models from these commonly used object detectors are evaluated from the perspectives of Meta-Architectures, Feature Extractors and Datasets. Similar to Huang et al. [33] who present multi-object detection primarily based on three recent meta architectures: SSD, Faster R-CNN and R-FCN. These meta architectures can combine with different feature extractors such as Inception, MobileNet and ResNet. Therefore, three common object detectors (1) SSD [51], (2) Faster R-CNN [66], and (3) R-FCN [16] are combined with three feature extractors (1) Inception V2 [74], (2) ResNet 101 [27], and (3) MobileNet V1 [32] to form standard object detectors:

1. SSD Inception V2
2. SSD MobileNet V1
3. R-FCN ResNet 101
4. Faster R-CNN Inception V2

One of the transfer learning methods is to use a pre-trained model from a first task to a second task on a separate dataset. Thus, we can take advantage of a pre-trained model by initializing the network with a set of pre-trained weights and biases from one task and re-train these parameters for a different task. Since one of our goals is to train general purpose detectors, i.e.: SSD, Faster R-CNN, R-FCN, and apply transfer learning to seek if these detectors perform well as dedicated multi-face detectors. These commonly cited detectors trained for multi-face detections can be used for the input of our tracking-by-detection approach. In our experiment, the models of SSD, R-FCN and Faster R-CNN have been trained on the COCO dataset. COCO dataset is a large scale object detection dataset where 80 object categories are given [33]. However, the COCO dataset does not contain any relevant face category, so we still re-train all layers in the detectors with a face dataset from the pre-trained COCO detector models. Using the pre-trained COCO detector models with their weights and biases for training multi-face detector models with the WIDER FACE or FDDB accelerate the fitting of multi-face detector models. As a result, the weights and biases trained on COCO dataset are changed such that the detector learns features specifically from the WIDER FACE or FDDB dataset. Hence, we obtain our multi-face detectors through this transfer learning technique. Overall, these detectors are trained on the WIDER FACE [87] dataset and FDDB [35] dataset to be used as multi-face detectors:

- MTCNN is trained on WIDER FACE
- SSD Inception V2 is trained on WIDER FACE
- SSD Inception V2 again is trained on FDDB
- R-FCN ResNet 101 is trained on WIDER FACE
- SSD MobileNet V1 is trained on FDDB
- Faster R-CNN Inception V2 is trained on WIDER FACE

The most suitable detectors from these 6 detectors will be used as multi-face tracking inputs.

4.3 Multi-face Tracking System

The Deep SORT algorithm [83] originally proposed for pedestrian tracking was introduced in the previous chapter. Multi-face movements in unconstrained settings may have long-time occlusion. Kalman filter predictions increase the uncertainty associated with the face location. The physical distance, such as Mahalanobis distance, is prone to generate larger uncertainty because it dramatically decreases the projected track mean. Meanwhile, hand-crafted features are not sufficiently discriminative to re-identify faces across different camera shots. To solve the problem above, the matching cascade gives priority to more frequently seen objects to encode the notice of probability spread in the association likelihood. Therefore, our adapted Deep SORT based algorithm for face tracking can overcome the problem of occlusion.

Our overall multi-face detection and tracking algorithm is as follows:

1. Apply MTCNN face detector or common object detectors trained on faces to the faces in frames
2. Use Kalman Filter to predict the motion of the faces
3. Embed the CNN structure from Table 3.1 to obtain the feature for each face
4. Use the Mahalanobis distance from Eq.3.13 to obtain the distance cost matrix
5. Apply feature classifiers from Section 3.2.4 to distinguish face features and obtain the feature appearance space from Eq.3.15
6. Initialize the Track indices $T = \{1, ..., N\}$, and Detection indices $D = \{1, ..., M\}$, and Maximum age A_{max}

7. Compute the Cost matrix C = $[c_{i,j}]$ using Eq.3.17
8. Compute the Gate matrix B = $[b_{i,j}]$ using Eq.3.18
9. Initialize set of matches $\mathcal{M} \leftarrow \emptyset$
10. Initialize set of unmatched detections $\mathcal{U} \leftarrow \mathcal{D}$
11. For n $\in \{1, ..., A_{max}\}$ do
    - Select tracks by age $\mathcal{T}_n \leftarrow \{i \in \mathcal{T} \mid a_i = n\}$
    - $[x_{i,j}] \leftarrow$ min cost matching $(C, \mathcal{T}_n, \mathcal{U})$
    - $\mathcal{M} \leftarrow \mathcal{M} \cup \{i, j) \mid b_{i,j} * x_{i,j} > 0\}$
    - $\mathcal{U} \leftarrow \mathcal{U} \setminus \{j \mid \sum_i b_{i,j} * x_{i,j} > 0\}$
12. End Step 6
13. Return $\mathcal{M}, \mathcal{U}$
14. Display the optimized matched tracking

The steps 2, 4 and 6-14 are originally from the Deep SORT algorithm. We apply our investigation on face detectors to the trcade starts to create a set of tracks $\mathcal{T}$, a set of detection $\mathcal{D}$, the maximum aacking input and we use our face feature trained on YouTube Faces dataset, in addition to our integration of the Angular Softmax Classifier into the matching cascade to distinguish different faces. To begin, we use either the MTCNN face detector or an object detector trained on faces to locate faces in video frames. Afterwards, the Kalman Filter predicts the location of faces in the next frame and applies Mahalanobis distance to find the physical space between the current detection location and the predicted location. Then, the feature of faces is obtained from the CNN and applies feature classifiers to distinguish face features with their feature appearance space. The matching casge A_{max}, and initializes the set of matches and unmatched detection $\mathcal{M}, \mathcal{U}$. Next, the

matching cascade computes the cost matrix and gate matrix, then it iterates over track age n to obtain the minimum cost matrix by solving linear assignment problems for tracks of increasing age. Since there are tracks left over not associated with a detection in the last n frames, a linear assignment can be applied between tracks in $\mathcal{T}_n$ and unmatched detections $\mathcal{U}$. Finally, the overall set of matches and unmatched detections are updated, which results in optimal matches across frames.

4.4 Summary

In this chapter, we provide our proposed methods of multi-face detection and tracking based on the Deep SORT matching cascade with our adaptation on faces and face detection add-ons with our investigation on face detectors in addition to the Angular Softmax Classifier embedded in the Deep SORT matching cascade.

In next chapter, we present the experimental methods for multi-face detection based on detectors in this chapter, and the multi-face tracking system based on deep feature association metrics with different feature classifiers

Chapter 5

Experiments

The experiments are conducted in two phases: Multi-face detection and multi-face tracking. In the multi-face detection phase, MTCNN is compared to three common object detectors. The multi-face tracking phase has four parts: (1) Compare Deep SORT face adaptation with two state-of-the-art offline multi-face trackers [49] [93] and two online trackers [37] [9]. (2) Compare four different classifier loss functions with each other in our multi-face tracking system. (3) Compare different detection methods as tracking inputs and evaluate the effects from detection. (4) Evaluate our adapted Deep SORT for faces on our new benchmark videos, and compare it to IOU-Tracker [9].

The hardware in our experiment is an Intel Core i7-7770k CPU with 16 GB system memory and 1 TB Hard Disk Drive, and the grahical adapter is a NVIDIA Geforce GTX1060. The software framework is Tensorflow version 1.12, developed by the Google Brain Team, which is an open source machine learning framework based on data flow graphs computing [90]. It is also adapted by the NVIDIA proprietary driver and CUDA toolkit with CUDNN installed to support GPU accelerated computation.

5.1 Multi-Face Detection Tasks

In this section, we present the implementation of MTCNN, SSD, R-FCN and Faster R-CNN and their training procedures.

5.1.1 Training of MTCNN

Since MTCNN is one of the dedicated multi-face detectors, we reproduce the training progress of MTCNN on WIDER FACE dataset with the Tensorflow Framework. The procedures are listed below:

1. We obtain the WIDER FACE training set and CelebA landmark training set from their official websites.
2. We generate and merge training and landmark data together for P-Net, described in Section 3.1.3, and we convert the data into TFRecord format which stores a sequence of binary records, then we train the P-Net.
3. We repeat the second step, in addition to generating TFRecord of negative, positive, parts and landmarks for R-Net, then we train the R-Net.
4. We repeat the third step to generate TFRecord of negative, positive, parts and landmarks for O-Net. Then, we train the O-Net.

5.1.2 Multi-Face Training Tasks

The following are the training procedures for our multi-face detector from pre-trained SSD, R-FCN and Faster R-CNN:

1. The first step is to pre-process data by converting WIDER FACE and FDDB to the format of VOC Pascal benchmark since it contains the label file bounding box values y_{max}, x_{max}, y_{min}, x_{min}.

2. The second step is to create indexes from the dataset in the VOC Pascal format file to a CSV file to store the annotation data.

3. The last step is to generate the training set and validation set into the TF Record format.

5.1.3 Multi-face Detection Training Criteria

We implement all the detectors models on a GeForce GTX1060 GPU. The training results are presented based on the following eight evaluation criteria [33]:

1. The Average Precision at 0.50 IOU represents the area under the precision-recall curve at 50% intersection over union between the predicted bounding boxes and ground truth bounding boxes.

2. The Average Precision at 0.75 IOU represents the area under the precision-recall curve at 75% intersection over union between the predicted bounding boxes and ground truth bounding boxes.

3. The SSD Classification Loss uses softmax loss which is the same as in Equation 3.19.

4. The SSD Localization Loss calculates the $smoothL1$ loss between the width/height of the ground truth bounding boxes and the width/height of the predicted bounding boxes. Smooth L1 is calculated in the following equation:

$$smoothL1(x) = f(n) = \begin{cases} 0.5x^2, & \text{if } \mid x \mid < 1 \\ \mid x \mid -0.5, & \text{otherwise} \end{cases} \tag{5.1}$$

where x is the distance between ground truth bounding box location and detection bounding box location. When $\mid x \mid$ is larger than 1, the output of $smoothL1$ is bigger than 0.5.

The localization loss between the predicted box l and the ground truth box g is defined [51]:

$$L_{loc}(x, l, g) = \sum_{i \in Pos}^{N} \sum_{m \in cx,cy,w,h} x_{i,j}^{k} smoothL1(l_i^m - \hat{g}_j^m), \tag{5.2}$$

where cx, cy are the offset to the default bounding box d of width w and height h: $\hat{g}_j^{cx} = (g_j^{cx} - d_i^{cx})/d_i^w$, $\hat{g}_j^{cy} = (g_j^{cy} - d_i^{cy})/d_i^h$, $\hat{g}_j^w = log\frac{g_j^w}{g_i^w}$, $\hat{g}_j^h = log\frac{g_j^h}{g_i^h}$. The variable $x_{i,j}^k$ is defined as:

$$x_{i,j}^k = \begin{cases} 1, & \text{if IoU} > 0.5 \text{ between default box i and ground truth j on class k} \\ 0, & \text{otherwise} \end{cases} \tag{5.3}$$

5. The SSD Regularization Loss indicates sum of weights in functions between the predicted result and ground truth. SSD uses L1 regularization as follows:

$$L_1 = \lambda \sum_i |\omega_i|, \tag{5.4}$$

where λ is the penalty term which determines how much to penalize the weights.

6. The Box Classifier Classification Loss is the softmax function for classifying the object, which is the same to Equation 3.19.

7. The Box Classifier Localization Loss is the function used to calculate the bounding box location regression, defined as:

$$L_{reg}(t_i, t_i^*) = R(t_i - t_i^*), \tag{5.5}$$

where R is the robust loss function from $smoothL1$ [21], and parameterizations t_i (predicted results) and t_i^* (ground truth) based on box's center coordinates x, y, w, and h:

$t_x = (x - x_a)/w_a$, $t_y = (y - y_a)/h_a$, $t_w = log(w/w_a)$, $t_h = log(h/h_a)$, $t_x^* = (x^* - x_a)/w_a$, $t_y^* = (y^* - y_a)/h_a$, $t_w^* = log(w^*/w_a)$, $t_h^* = log(h^*/h_a)$,

where variables x, x_a, and x^* are the predicted box, anchor box, and ground truth bounding box, respectively, and are in the same format for y, w, and h [66].

8. The RPN Localization Loss calculates the regression loss between the anchor box and ground truth bounding box using Equation 5.5 above.

5.2 Multi-Face Tracking Tasks

For long term multi-face tracking [37] [63] in unconstrained videos, locating each face and successfully tracking them over time while the face leaves and re-enters the camera are complex tasks. Online trackers demonstrate good performance in constrained videos, while these videos are produced with very stationary and slow-moving cameras. We evaluate our adapted Deep SORT for faces which is an online tracker in unconstrained videos and compare its performance with two state-of-the-art offline multi-face trackers, and two online trackers.

5.2.1 Training of Feature Classifiers

The feature classifiers in the multi-face tracking system need to be obtained by training the deep feature CNN followed with classifier loss functions. First, is the pre-processing step for the YouTube Faces dataset, we randomly select faces from 700 different celebrities' faces as the training set and randomly select another 700 celebrities' faces from the rest as the testing set, and we split off 10% of the training data for validation. Each face ID

consists of 1 to 6 clips. We select the first 15 frames in each clip. This step ensures the diversity of faces and minimizes the training effort. Second, we crop out the faces from the given ground-truth bounding boxes. Third, we label faces ID from 0 to 699 for both training and testing set based on their names alphabetically. Finally we train four feature classifier loss functions which we have introduced in Section 3.2.4: (1) Cosine Softmax Classifier, (2) Angular Softmax Loss, (3) Magnet Loss, (4) Triplet Loss on training set and testing set in CNN network presented on Table 3.1.

5.2.2 Implementation and Evaluation of Adapted Deep SORT

Since the basic SORT cannot handle faces tracking in unconstrained environments, we primarily use the face feature distance rather than spatial distance. Therefore, we set λ as zero in Equation 3.17. As a result, only the feature appearance information is used in the association cost term. Thus, we only take the feature loss metric. The original Deep SORT algorithm is a general MOT tracker and has been previously used in pedestrian tracking. It demonstrates good performance on MOT evaluation metrics. We embed four different classifier loss functions for faces into the adapted Deep SORT for faces to conduct multi-face tracking via features. We apply the well-performing detectors from the multi-face detection tasks and use the ground-truth detections as the input for multi-face tracking. For data processing of the detection result: First, we extract the frames of the music videos. Second, we store each sequence of frames of a video into a separate binary file. Each file contains an array of shape N x 138, where N is the number of detections in the corresponding face sequence. The 138 parameters represent 10 columns of detection information in the detection format and 128 dimensional output feature. Third, we use the four different feature classifier loss functions in the matching cascade to evaluate multi-face tracking performance based on their feature distance metric losses. Finally, we fine-tune the maximum age for detection input in each video clip. We compare the various versions of our adapted Deep SORT for faces are compared with two state-of-the-art offline trackers:

(1) the tracker by Lin et al. [49], (2) the ADF tracker [93], and two online trackers: mTLD [37] and IOU-Tracker [9]. The IOU-Tracker uses our detectors for tracking inputs.

5.2.3 Multi-face Tracking Evaluation Criteria

In the MOT challenge benchmark, MOT performance is evaluated based on metrics such as Recall, Precision, Mostly Track (MT), Identity Switches (IDS), Fragment (FRAG), MOTA, MOTP, etc [6]. Same as MOT, our evaluation of multi-face tracking is based on the following metrics [59]:

1. Recall: Fraction of objects that are correctly tracked:

$$Recall = TP_i/(TP_i + FP_i), \tag{5.6}$$

   where TP_i is the true positive of object i, and FP_i is the false positive of object i.

2. Precision: Fraction of correct objects to all objects that are tracked.

$$Precision = TP_i/(TP_i + FN_i), \tag{5.7}$$

   similar to the Recall equation, while FN_i (false negative) of object i replaces FP_i.

3. Mostly tracked (MT): Percentage of ground-truth tracks that have the same label for at least 80% of their life span.

4. Identity switches (IDS): Number of times the reported identity of a ground-truth track changes.

5. Fragmentation (FM): Number of times a track is interrupted by a missing detection.

6. Multi-object tracking accuracy (MOTA): Summary of overall tracking accuracy in terms of false positive, false negative, ID switches and ground truth. Defined in Equation below:

$$MOTA = (1 - \frac{FN + FP + IDS}{GT}) * 100\% \in (-\infty, 100\%]. \tag{5.8}$$

7. Multi-object tracking precision (MOTP): Summary of overall tracking precision in terms of bounding box overlap between ground-truth and reported location.

$$MOTP = \frac{\sum_{t,i} d_{t,i}}{\sum_t c_t}, \tag{5.9}$$

where c_t denotes the number of matches in frame t, and $d_{t,i}$ is the bounding box overlap between the predicted object i and its ground truth.

5.3 Summary

In this chapter, we presented our experimental methods for multi-face detection and multi-face tracking. We introduced the dataset for training, testing and validation of multi-face detectors. We presented feature classifiers for multi-face tracking and the benchmarks for evaluations of multi-face tracking. We also presented multi-face detection training criteria and multi-face tracking evaluation criteria.

In the next chapter, we present training results of multi-face detection and evaluation results of multi-face tracking. Then, we discuss the evaluation results by comparing to different trackers.

Chapter 6

Results and Analysis

In this chapter, we first present the results of multi-face detection and give a full analysis prior to proceeding the multi-face tracking, since the input of multi-face tracking is highly dependent on multi-face detection. Moreover, we demonstrate the evaluation results of multi-face tracking performed on the Music Video Dataset and on our new benchmark.

6.1 Multi-Face Detection Results and Analysis

6.1.1 Detection Training Results

First, we present all detectors (MTCNN, SSD, Faster R-CNN and R-FCN) in both, Average Precision at 0.50 IOU and 0.75 IOU separately in Figure 6.1 and 6.2, respectively. Second, the SSD localization loss from Equation 5.3 and box classifier localization loss for R-FCN and Faster R-CNN from Equation 5.5 calculate localization loss of an object bounding box identically, i.e., they are compared and shown in Figure 6.3. Third, because all of our common object detectors calculate classification loss using the softmax loss function, they are compared and shown in Figure 6.4. Fourth, the regularization loss of SSD is shown in Figure 6.5. Finally, R-FCN and Faster R-CNN are two-stage object detectors. The

training in the first stage for RPN localization loss is visualized in Figure 6.6.

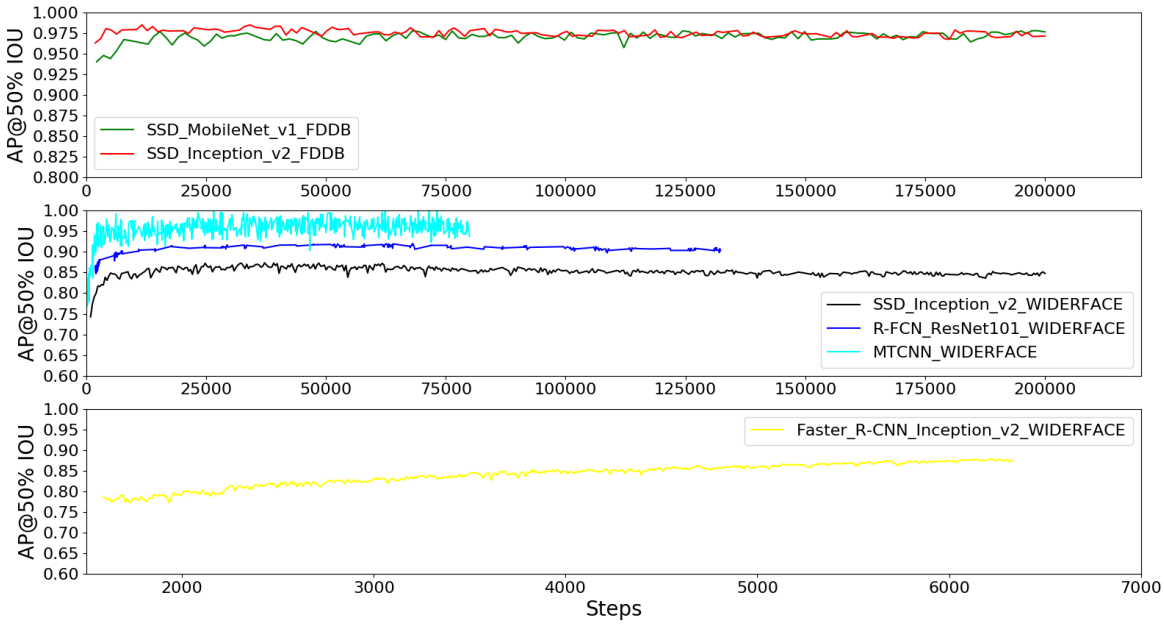

Figure 6.1: Average Precision at 0.50 IOU

6.1.2 Detection Results Analysis

The results from these common object detectors (R-FCN, Faster R-CNN and SSD) are compared based on the respective dataset used for training, either on FDDB or WIDER FACE. The testing and evaluation datasets by themselves, and the faces are different between FDDB and WIDER FACE. Since most of the videos from the benchmark Music Video dataset are unconstrained videos, they are hard examples to detect with these common detectors. Meanwhile, the WIDER FACE dataset compared to FDDB dataset consists of a greater amount of and more diverse faces. Figure 6.1 indicates the Average Precision (AP) at 50% IOU: both SSD Mobilenet V1 trained on the FDDB dataset and SSD Inception V2 trained on the FDDB dataset have achieved 97.70% and 97.23%, respectively. The detector models: MTCNN, SSD Inception V2, Faster R-CNN Inception V2

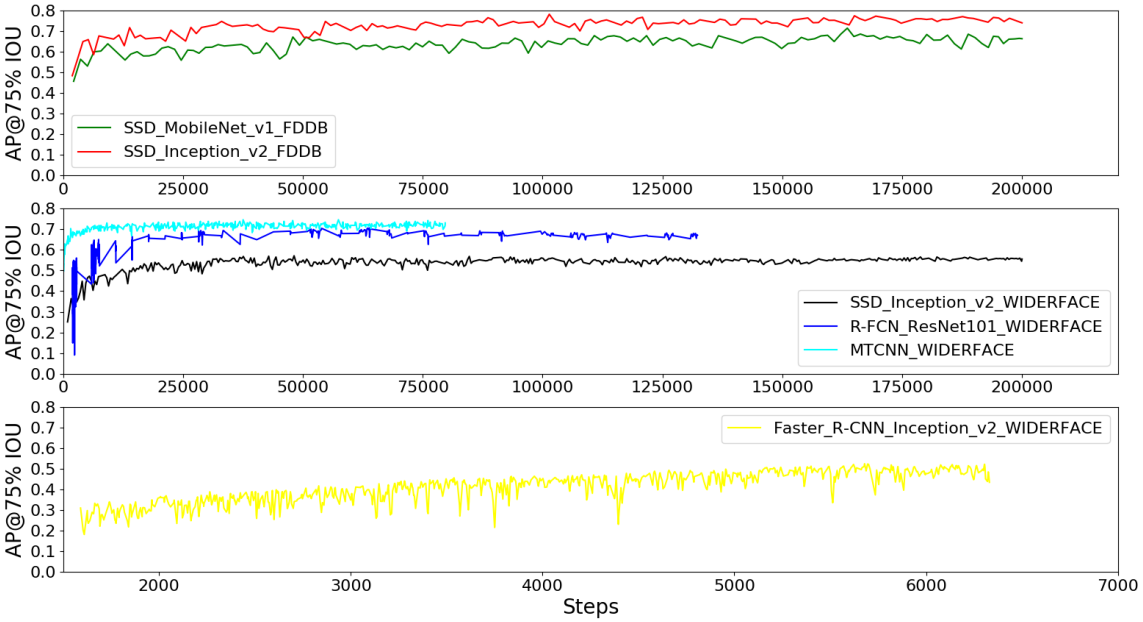

Figure 6.2: Average Precision of 0.75 IOU

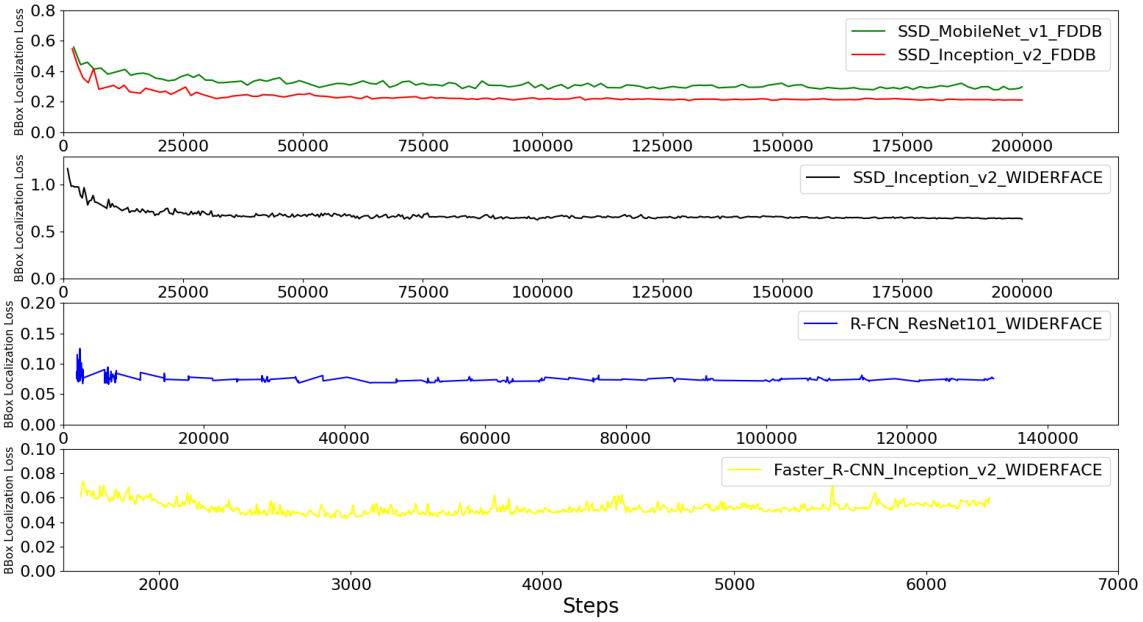

Figure 6.3: Bounding Box Localization Loss

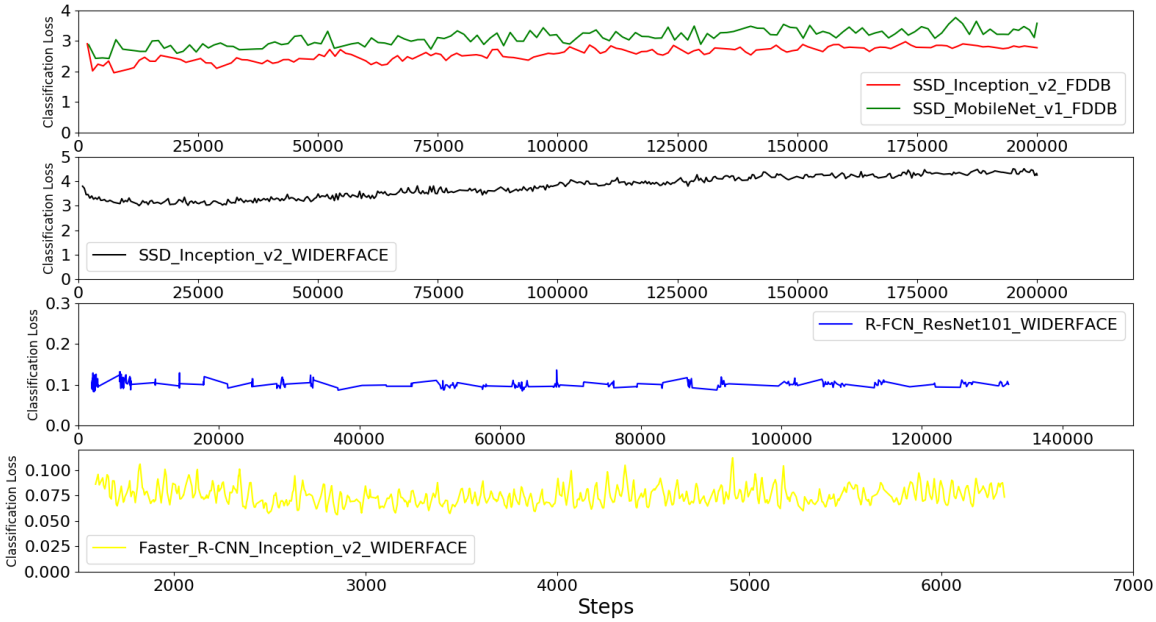

Figure 6.4: Classification Loss

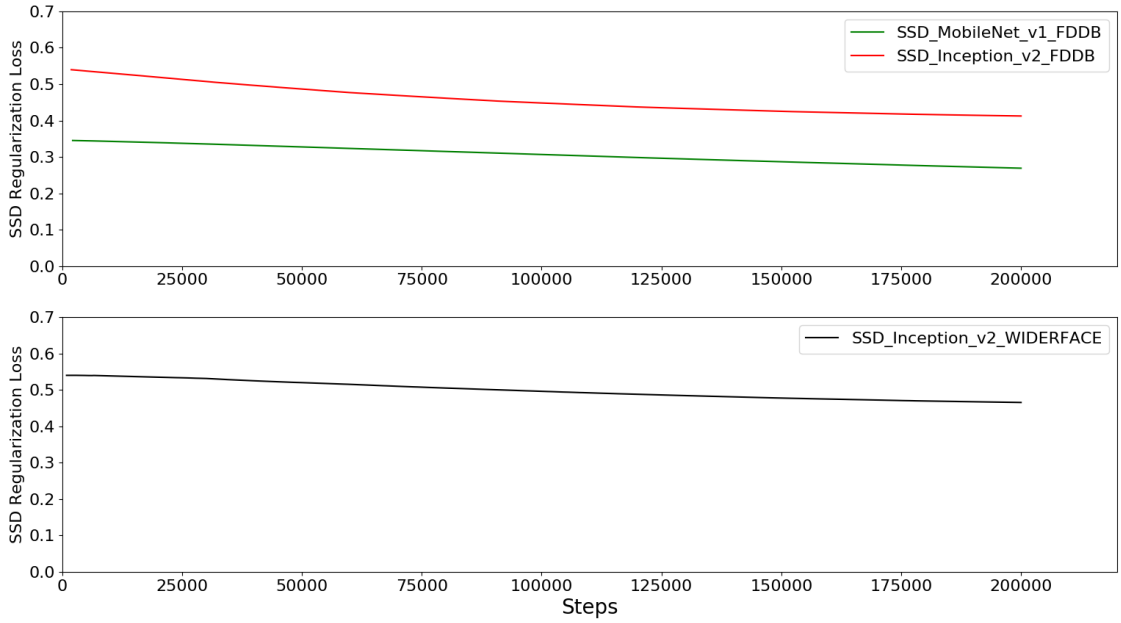

Figure 6.5: SSD Regularization Loss

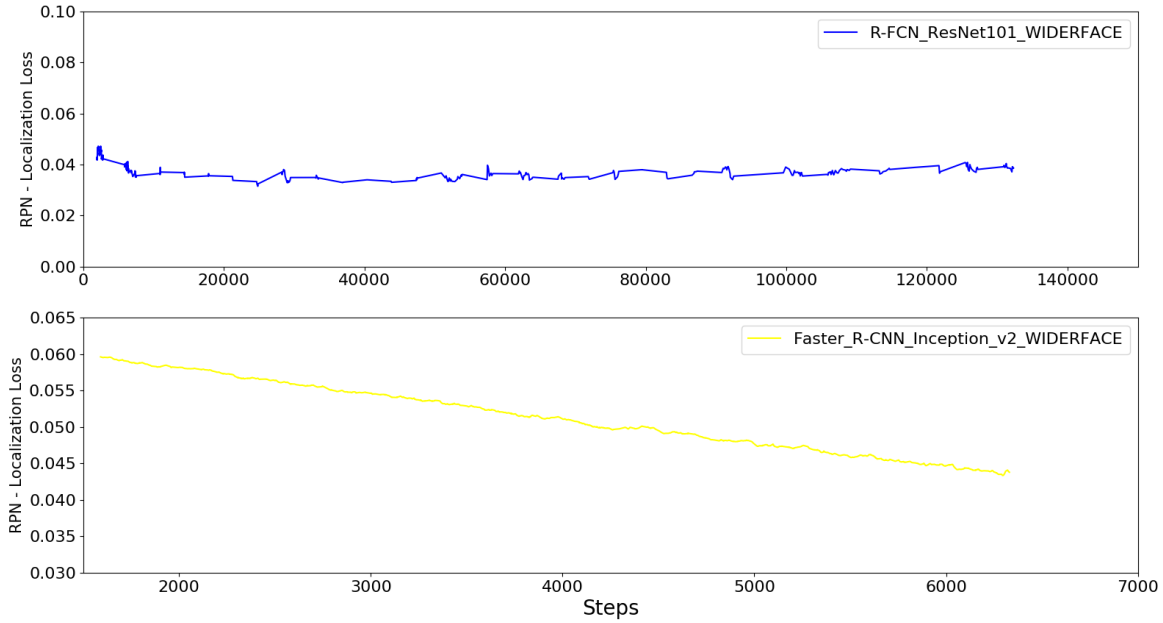

Figure 6.6: RPN Localization Loss

and R-FCN ResNet 101 are trained on the WIDER FACE dataset. Of all detectors trained on WIDER FACE, MTCNN achieved the highest average precision. R-FCN ResNet 101 achieved the second highest average precision. However, training Faster R-CNN Inception V2 on WIDER FACE was very costly, and it took 36 days on our available hardware for only 6,325 steps. Since Faster R-CNN is an incomplete training model and has a lower precision results than trained models of MTCNN and R-FCN, it will not be used as the input for the tracking system.

Figure 6.2 shows the average precision at 75% IOU. Compared to the AP at 50% IOU, 75% IOU requires a larger intersection over union to be matched. Hence, each result is lower than the corresponding result shown in the AP 50% IOU. We observe that average precisions of SSD Inception V2 and SSD MobileNet V1 at 50% IOU are very close in Figure 6.1. However, Figure 6.2 reveals that SSD Inception V2 achieved higher average precision at 75% IOU than SSD MobileNet V1. The R-FCN training curve reveals that there are

gaps between datapoints, so we look up the CSV file of R-FCN average precision versus steps, and we observe that the Tensorboard stored more data of average precision versus steps at the beginning of the training and less data when the steps increase. That is why the plot result is dense near beginning of the training, then it becomes smoother when steps increase.

Figure 6.3 indicates SSD Inception V2's predicted detection bounding box has a lower loss than SSD MobileNet V1 because MobileNet V1 uses the ReLU activation function, and this nonlinear bottleneck results in feature loss in low level feature. SSD Inception v2 trained on WIDER FACE suggests that a steady localization loss is performed, and it converges at the end of training. Meanwhile, the computational cost in the first stage for Faster R-CNN is very expensive but losses of R-FCN and Faster R-CNN are similar.

Figure 6.4 shows mild overfitting occuring during the training progress as the curves of SSD Inception V2 trained on both FDDB and WIDER FACE, and the curve of SSD MobileNet V1 trained on FDDB increase. It reveals that the training should be stopped around 25000 steps while more training increases the training time.

Figure 6.5 shows that SSD MobileNet V1 has less regularization loss than SSD Inception V2 because the backbone of MobileNet is simpler and more shallow than Inception V2. Therefore, MobileNet has lower number of parameters with less number of weights that need to regularized. As a result, the regularization loss of SSD MobileNet V1 has lower values than the regularization loss of SSD MobileNet V2.

Figure 6.6 demonstrates that R-FCN has lower localization loss in the first stage. However, R-FCN has mild overfitting occurring after 20000 steps while Faster R-CNN has loss curve decreasing from beginning to the point when training was stopped.

Overall, MTCNN achieves the highest average precision of these detectors trained on WIDER FACE, and R-FCN achieves the second highest average precision. Also, R-FCN achieves the highest average precision among the common object detectors trained on

Detectors	Average Precision at 50% IOU	Average Precision at 75% IOU
MTCNN	96.50	73.04
R-FCN ResNet 101	90.31	67.10
SSD Inception V2	84.76	55.55
Faster R-CNN Inception V2	87.42	48.18

Table 6.1: Average Precision Results of Detectors trained on WIDER FACE

Detectors	Average Precision at 50% IOU	Average Precision at 75% IOU
SSD Inception V2	97.23	76.22
SSD MobileNet V1	97.70	66.33

Table 6.2: Average Precision Results of SSD trained on FDDB

WIDER FACE. SSD Inception V2 achieves better average precision from Figure 6.2 and lower losses based on Figure 6.3 and 6.4. Therefore, we use MTCNN for one detection stage of our tracker and R-FCN as a second detector for the tracking system, we also select SSD Inception V2 trained on FDDB as the third detector.

Table 6.1 shows the results of average precision at 50% IOU and 75% IOU from the detectors trained on WIDER FACE, and Table 6.2 shows the results of average precision at 50% IOU and 75% IOU from SSD trained on FDDB.

6.2 Multi-Face Tracking Results and Analysis

In this section, we present evaluation results of our adapted Deep SORT for faces on the Music Video Dataset and the House of Commons Dataset, and we compare our adapted Deep SORT with two state-of-the-art offline multi-face trackers [49][93] and two online trackers [37] [9].

6.2.1 Evaluation Results on the Music Video Dataset

In the Music Video Evaluation Table 6.3, "SSD" is the SSD Inception V2 detector trained on the FDDB dataset, "R-FCN" is the R-FCN ResNet 101 detector trained on the WIDER

Methods	Recall↑	Precision↑	MT↑	IDS↓	FRAG↓	MOTA↑	MOTP↑
Lin et al. [49]	81.7	90.2	32	624	1645	69.2	86.0
ADF-mTLD [93]	69.1	88.1	14	1914	2786	57.7	80.1
ADF-Siamese [93]	71.5	89.4	18	986	2512	62.3	64.0
ADF-Triplet [93]	71.8	88.8	19	902	2546	61.8	64.2
ADF-SymTriplet [93]	71.8	89.7	19	699	2563	62.8	63.2
ADF-SymTriplet-Contx [93]	73.2	90.5	19	625	2417	64.1	64.2
mTLD [37]	4.6	21.5	0	192	453	-8.2	69.1
IOU-Tracker(SSD) [9]	11.6	85.9	0	551	777	9.1	36.5
IOU-Tracker(R-FCN) [9]	46.4	82.3	7	2674	3325	33.6	33.6
IOU-Tracker(MTCNN) [9]	80.1	80.0	28	3792	4034	56.1	32.8
Deep SORT-Angular(SSD)	11.5	82.3	0	330	752	8.7	36.4
Deep SORT-Cosine(SSD)	11.4	82.5	0	345	748	8.6	36.4
Deep SORT-Triplet(SSD)	11.4	82.2	0	326	750	8.6	36.4
Deep SORT-Magnet(SSD)	11.4	82.5	0	318	732	8.6	36.4
Deep SORT-Angular(R-FCN)	48.9	80.7	5	1477	2903	36.5	36.8
Deep SORT-Cosine(R-FCN)	47.3	81.6	4	1402	2917	34.0	33.8
Deep SORT-Triplet(R-FCN)	43.9	81.2	5	1428	3120	33.2	33.8
Deep SORT-Magnet(R-FCN)	43.7	81.3	4	1415	3134	32.8	33.9
Deep SORT-Angular(MTCNN)	79.3	77.0	28	2408	4476	53.0	33.1
Deep SORT-Cosine(MTCNN)	79.0	77.2	28	2409	4553	53.1	33.1
Deep SORT-Triplet(MTCNN)	79.2	76.8	28	2413	4518	52.7	33.1
Deep SORT-Magnet(MTCNN)	79.3	79.6	28	2386	4488	52.9	33.1
Deep SORT-Angular(*)	95.0	96.1	50	3051	2214	87.9	8.2
Deep SORT-Cosine(*)	94.7	96.2	50	2967	2339	87.8	8.2
Deep SORT-Triple(*)	94.8	96.1	50	3068	2282	87.7	8.2
Deep SORT-Magnet(*)	94.8	96.1	50	3001	2302	87.8	8.2

Table 6.3: Music Video Evaluation

FACE dataset, and MTCNN is trained on the WIDER FACE dataset. The Deep ORT(*) methods listed at the bottom of Table 6.3 use perfect detection from ground-truth detection results, which brings approximately zero error for the detection stage. We observe that MTCNN is the best detector based on evaluation metrics of Recall, MT, MOTA and MTCNN for which adapted Deep SORT for faces works well. Lin et al. [49] use body parts detector as the co-occurrence model and ADF by Zhang et al. [93] use a famous face detector by Mathias et al. [58]. Therefore, the recall, precision values of their state-of-the-art multi-face offline trackers which are ADF [93], and Lin et al [49] are higher than ours. We observe that SSD could not track one face in 80% of its life span based on MT criteria. Hence, our two detectors trained on FDDB are not able to work in a challenging environment such as Music Video Dataset. Perfect detection input leads to

the highest IDS because more faces are detected which leads to more faces for tracking, thus more faces are switching their identities. The IOU-Tracker achieves the online and real-time multi-object tracking without using image information. We observe that IOU-Tracker's evaluation results in Recall, Precision, MT, MOTA, MOTP are roughly the same as the ones by our adapted Deep SORT for faces. However, our adapted Deep SORT method for faces is significantly better than the IOU-Tracker in the IDS criteria because our discriminative feature classifiers have capability to distinguish different face IDs within the number of max-age frames. The IOU-Tracker is not using any image information. Therefore, without keeping face feature in the tracker, the IOU-Tracker frequently changes face IDs.

When comparing the performance of four feature classifier loss functions in our adapted Deep SORT matching cascade for faces, we find that the Angular Softmax Classifier is slightly better than the other classifier loss functions. When we use perfect detection as the tracking input, we observe that the fragment (FRAG) value in Angular Softmax is approximately half of the loss of other classifiers. Despite the fact that IDS in our method is still higher than for the ADF tracker [93] and Lin et al. [49], the Angular Softmax gives less fragments in comparison to them.

Figure 6.7 shows successful examples of tracking on the Music Video Dataset. We observe that the adapted Deep SORT tracker for faces can track and identify the same face under conditions of changing face scales, different sides of faces, with or without glasses, and face occlusions. Conversely, Figure 6.8 gives poor examples due to long time gap between similar faces, low resolutions of video frames and dark background. Both good results and poor results use the adapted Deep SORT method with the Angular Loss and MTCNN detector for the input of the tracker.

Zhang et al. [93] indicate that "Third, the CNN fine-tuning process is time-consuming. It takes around 1 hour on a NVIDIA GT980Ti GPU for 10,000 back propagation iterations". Meanwhile, Lin et al. [49] state that "In one 5-minutes music video, there are

Methods	Speed(fps)	Methods	Speed(fps)
mDeepSORT-Angular	194.7	mDeepSORT-Magnet	184.6
mDeepSORT-Cosine	171.9	mDeepSORT-Triplet	172.5

Table 6.4: Multi-Face Tracking Speed without Detection on Music Video

21,747 faces observations over a sequence of 5,000 frames, our implementation takes about 25 minutes after feeding the detection results." Therefore, their methods need to process the videos offline for tracking, and they cannot be applied in real-time tracking. Table 6.4 shows that our adapted Deep SORT multi-face tracking achieves online and real-time tracking.

Likewise, mTLD is an online tracker by learning updated detection using a Nearest Neighbor classifier. The Deep SORT association metric performs better than mTLD. Our adapted Deep SORT, mTLD and IOU Tracker all satisfy online and real-time tracking speed requirements.

6.2.2 Evaluation and Comparison on our benchmark – The House of Commons

We conduct the evaluation of our adapted Deep SORT method for faces on our new benchmark – The House of Commons, and we compare our adapted Deep SORT method for faces with the IOU-Tracker using our detection input from SSD, R-FCN and MTCNN detectors. The evaluation results on three clips: QPA [11] , BOIE [10] and HOCA [12] are shown in Table 6.5, 6.6 and 6.7, respectively. The evaluation results on the entire benchmark including all 7 clips is shown in Table 6.8.

The evaluation results on our new benchmark in Table 6.5, 6.6 and 6.7 also reveal the same behavior than from evaluations on the Music Video dataset: The results of our adapted Deep SORT method for faces evaluated on our new benchmark are roughly the same as the results of the IOU-Tracker in following evaluation metrics: Recall, Precision,

Methods	Recall↑	Precision↑	MT↑	IDS↓	FRAG↓	MOTA↑	MOTP↑
IOU-Tracker(SSD)	2.4	100.0	2	2	3	2.3	31.3
IOU-Tracker(R-FCN)	2.8	91.9	2	3	4	2.5	13.3
IOU-Tracker(MTCNN)	70.2	98.0	12	189	192	66.0	25.3
DeepSORT-Angular(SSD)	2.4	100.0	2	0	4	2.4	34.0
DeepSORT-Cosine(SSD)	2.4	100.0	2	0	4	2.4	34.0
DeepSORT-Magnet(SSD)	2.4	100.0	2	0	4	2.4	34.0
DeepSORT-Triplet(SSD)	2.4	100.0	2	0	4	2.4	34.0
DeepSORT-Angular(R-FCN)	2.8	93.7	2	1	5	2.6	13.6
DeepSORT-Cosine(R-FCN)	2.8	93.7	2	1	5	2.6	13.6
DeepSORT-Magnet(R-FCN)	2.8	93.7	2	1	5	2.6	13.6
DeepSORT-Triplet(R-FCN)	2.8	94.6	2	1	5	2.7	13.6
DeepSORT-Angular(MTCNN)	74.0	97.1	13	41	133	71.2	24.9
DeepSORT-Cosine(MTCNN)	73.8	97.2	13	41	130	71.0	24.9
DeepSORT-Triplet(MTCNN)	74.0	97.1	13	41	133	71.2	24.9
DeepSORT-Magnet(MTCNN)	73.9	97.1	13	39	133	71.1	24.9

Table 6.5: QPA Video Evaluation

Methods	Recall↑	Precision↑	MT↑	IDS↓	FRAG↓	MOTA↑	MOTP↑
IOU-Tracker(SSD)	13.7	100.0	1	26	37	13.0	31.6
IOU-Tracker(R-FCN)	74.7	88.3	10	77	13	63.0	14.1
IOU-Tracker(MTCNN)	86.4	98.6	15	91	25	82.9	18.0
DeepSORT-Angular(SSD)	14.5	99.7	1	2	36	14.4	31.3
DeepSORT-Cosine(SSD)	14.3	99.7	1	2	34	14.2	31.3
DeepSORT-Magnet(SSD)	14.4	99.7	1	2	36	14.3	31.3
DeepSORT-Triplet(SSD)	14.5	99.7	1	2	36	14.4	31.3
DeepSORT-Angular(R-FCN)	74.5	87.7	10	76	16	62.2	14.3
DeepSORT-Cosine(R-FCN)	74.4	87.7	10	76	19	62.1	14.3
DeepSORT-Magnet(R-FCN)	74.5	87.8	10	75	18	62.3	14.3
DeepSORT-Triplet(R-FCN)	74.5	87.8	10	76	17	62.3	14.3
DeepSORT-Angular(MTCNN)	86.2	97.6	15	75	33	82.2	18.0
DeepSORT-Cosine(MTCNN)	86.7	97.4	15	74	35	82.5	18.1
DeepSORT-Triplet(MTCNN)	86.4	97.5	15	75	33	82.4	18.0
DeepSORT-Magnet(MTCNN)	87.0	97.4	15	77	39	82.2	18.1

Table 6.6: BOIE Video Evaluation

MT, MOTA and MOTP. Intuitively, our adapted Deep SORT method for faces achieves lower IDS compared to IOU-Tracker's IDS. SSD could not be used for detection of both our adapted Deep SORT and IOU-Tracker in HOCA clip because it only detects one face within two frames, so the MOT evaluation metrics give NaN% due to the extremely low detection rate.

We also complete annotation of the entire House of Commons dataset, and we evaluate our adapted Deep SORT method on faces compared the IOU-Tracker on faces. Overall,

Methods	Recall↑	Precision↑	MT↑	IDS↓	FRAG↓	MOTA↑	MOTP↑
IOU-Tracker(R-FCN)	54.0	96.2	3	139	133	49.4	21.7
IOU-Tracker(MTCNN)	85.5	96.9	17	131	114	80.5	19.7
DeepSORT-Angular(R-FCN)	58.2	94.8	4	23	131	54.6	21.2
DeepSORT-Cosine(R-FCN)	58.2	94.9	4	20	129	54.7	21.2
DeepSORT-Magnet(R-FCN)	58.0	94.8	4	19	129	54.5	21.2
DeepSORT-Triplet(R-FCN)	57.8	94.8	4	25	130	54.2	21.1
DeepSORT-Angular(MTCNN)	88.0	96.1	18	41	79	83.7	19.1
DeepSORT-Cosine(MTCNN)	87.7	96.5	18	42	77	83.7	19.1
DeepSORT-Triplet(MTCNN)	87.7	95.4	18	35	75	83.8	19.1
DeepSORT-Magnet(MTCNN)	88.0	97.4	18	36	77	84.2	19.1

Table 6.7: HOCA Video Evaluation

Methods	Recall↑	Precision↑	MT↑	IDS↓	FRAG↓	MOTA↑	MOTP↑
IOU-Tracker(R-FCN) [9]	33.2	96.1	40	329	261	31.3	16.2
IOU-Tracker(MTCNN) [9]	80.2	94.4	109	1514	1477	74.5	18.5
DeepSORT-Angular(R-FCN)	33.8	95.5	40	141	267	32.0	16.1
DeepSORT-Cosine(R-FCN)	33.8	95.5	41	137	261	32.0	16.1
DeepSORT-Triplet(R-FCN)	33.7	95.5	41	142	260	31.9	16.1
DeepSORT-Magnet(R-FCN)	33.7	95.5	41	124	263	31.9	16.1
DeepSORT-Angular(MTCNN)	79.6	94.5	114	337	1187	74.4	18.7
DeepSORT-Cosine(MTCNN)	79.8	94.3	114	338	1221	74.4	18.8
DeepSORT-Triplet(MTCNN)	79.7	94.5	113	324	1222	74.6	18.8
DeepSORT-Magnet(MTCNN)	79.7	94.5	114	311	1213	74.6	18.8

Table 6.8: House of Commons Evaluation

our adapted Deep SORT method on faces has slightly more MT and much lower IDS. SSD Inception V2 could not be used for detection in two clips that have extremely low detection rate due to the challenge of the House of Common dataset.

Figure 6.9 demonstrates successful examples of detection and tracking on the House of Commons Video Benchmark which we use the MTCNN detector as input for the tracker and the Angular Loss for face ID classification. The top two frames have 83 frames in between. It successfully tracks and identifies same faces between the two frames even though ID 19 was occluded by a hug and ID 4 is detected as a side face in the first frame. The middle two frames reveal that all faces are far away from the camera, and they have low resolution, but our adapted Deep SORT method successfully tracks and identifies the majority of faces except for ID 8 and ID 2. ID 15 and ID 9 are able to be tracked after rapid seating.

Figure 6.10 shows poor examples of detection and tracking on the House of Commons Video Benchmark. We also use the MTCNN detector as input for the tracker and the Angular Loss for face ID classification. First, the top two frames show our tracking failed due to large scale changes of faces and rapid switching of camera angles. Second, the middle two frames shows that only ID 11 is successfully tracked, while ID 25 switches to ID 123, and ID 53 was originally assigned to ID 5. In the two bottom frames, ID 7 in the frame of bottom left switches to ID 36 in the bottom right due to low exposure.

From the analysis above, although our multi-face detection and tracking system performs fairly well in general, it still produces tracking errors such as IDS.

Figure 6.7: Successful Detection and Tracking Examples on Music Video Dataset

Figure 6.8: Poor Detection and Tracking Examples on Music Video Dataset

Figure 6.9: Successful Detection and Tracking Examples on the House of Commons Video Benchmark

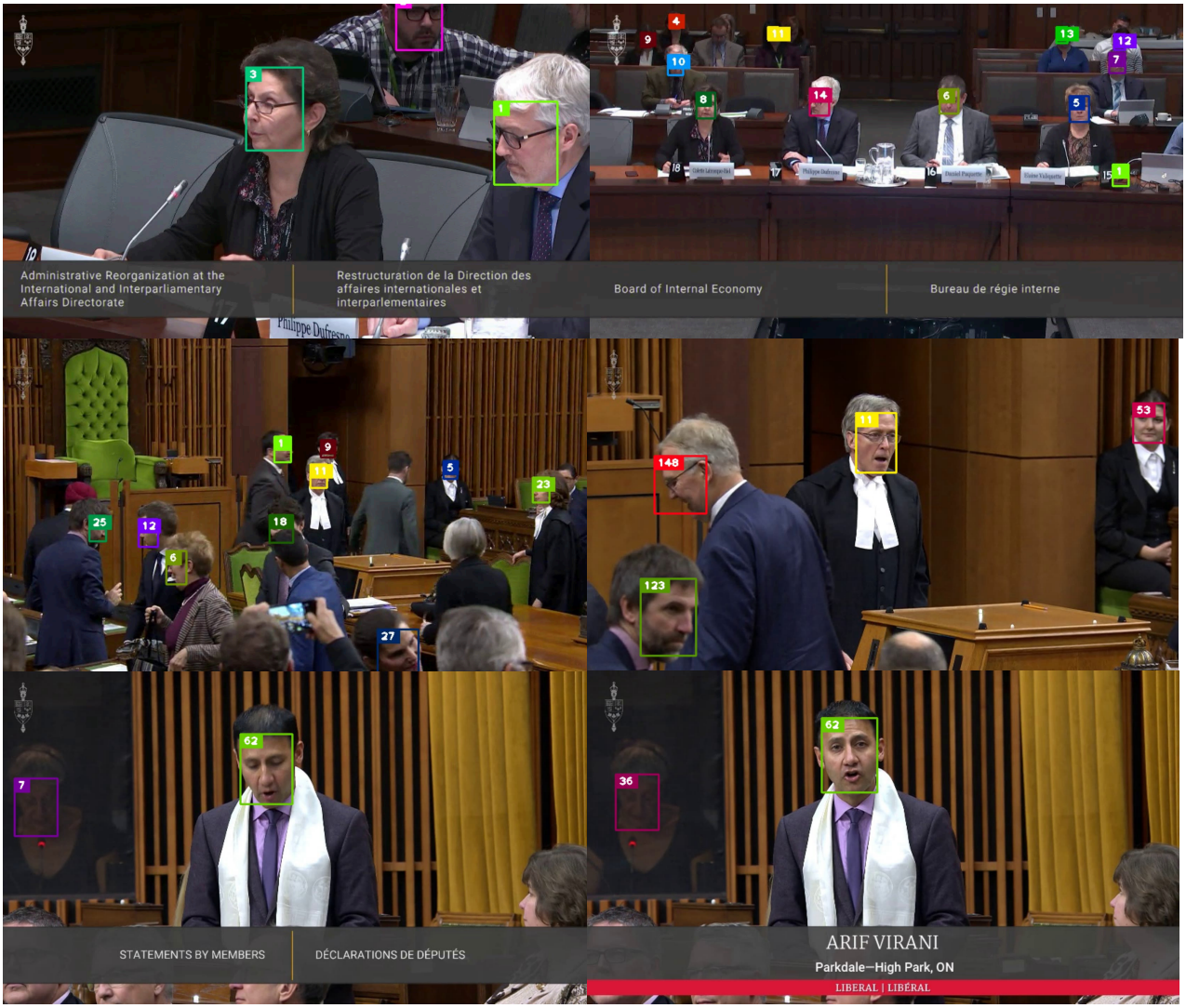

Figure 6.10: Poor Detection and Tracking Examples on the House of Commons Video Benchmark

Chapter 7

Conclusion and Future Work

7.1 Conclusion

Deep neural network techniques give untold possibilities for solving multi-face tracking problems. This thesis presents an online multi-face tracking approach based on our adapted Deep SORT matching cascade. We review and implement MTCNN, and three common object detectors with three different feature extractors on face detection, we also embed and compare four different feature classifier loss functions embedded in adapted Deep SORT feature association metrics. We compare our adapted Deep SORT multi-face tracking system with two state-of-the-art offline trackers and two online tracker by evaluating them on the Music Videos dataset. Finally, we contribute a new benchmark dataset for multi-face tracking, and we evaluate our adapted Deep SORT method and IOU-Tracker on this new benchmark.

Our adapted Deep SORT method has lower IDS compared to online trackers. Although our method still has high IDS compared to the state-of-the-art offline trackers, it still tracks most of the objects based on the MT results on the Music Video Dataset, and it is close to the state-of-the-art MT results from ADF trackers [93]. In addition, we achieved online and real-time tracking speed, and our method can be used for real-time tracking in a crowd

and fast-motion environment.

7.2 Limitations and Future Work

The results of MOTA, MOTP and IDS on the Music Video Dataset evaluation give us the feedback that we are limited by a well-discriminated feature classifiers for face re-identification. Compared to multi-vehicle and multi-pedestrian tracking, different vehicles or pedestrians have larger variations among themselves in appearance, such as different types of cars with different colors and then different pedestrians with different outfits in different colors. Conversely, different faces have smaller variations. Thus, making multi-face tracking is a difficult task to complete in unconstrained videos. We still need to build a better face detector for a well-performing tracking-by-detection input. Secondly, Lin et al. [49] develop a well-discriminated human-body classifier linked to a multi-face classifier to lower IDS, and increase MOTA and MOTP. Hence, we suggest to develop a human-body classifier to assist multi-face classification. The whole HoC dataset is now included.

References

[1] Stephen Balaban. Deep learning and face recognition: The state of the art. In *Biometric and Surveillance Technology for Human and Activity Identification XII*, volume 9457, page 94570B. International Society for Optics and Photonics, 2015.

[2] Behnam Banitalebi and Hamid Amiri. An improved nearest neighbor data association method for underwater multi-target tracking. In *IEEE workshop & exhibition on new trends for environmental monitoring using passive systems*, pages 1–4, 2008.

[3] Ankan Bansal, Anirudh Nanduri, Carlos D Castillo, Rajeev Ranjan, and Rama Chellappa. Umdfaces: An annotated face dataset for training deep networks. In *2017 IEEE International Joint Conference on Biometrics (IJCB)*, pages 464–473. IEEE, 2017.

[4] Tamer Basar. A new approach to linear filtering and prediction problems. 2001.

[5] Saman Bashbaghi, Eric Granger, Robert Sabourin, and Mostafa Parchami. Deep learning architectures for face recognition in video surveillance. In *Deep Learning in Object Detection and Recognition*, pages 133–154. Springer, 2019.

[6] Keni Bernardin and Rainer Stiefelhagen. Evaluating multiple object tracking performance: the clear mot metrics. *EURASIP Journal on Image and Video Processing*, 2008:1–10, 2008.

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

[8] Alex Bewley, Lionel Ott, Fabio Ramos, and Ben Upcroft. Alextrac: Affinity learning by exploring temporal reinforcement within association chains. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2212–2218, May 2016.

[9] Erik Bochinski, Volker Eiselein, and Thomas Sikora. High-speed tracking-by-detection without using image information. In *International Workshop on Traffic and Street Surveillance for Safety and Security at IEEE AVSS 2017*, Lecce, Italy, August 2017.

[10] Canada, Parliment, and House of Commons. 43th parliament, board of internal economy. https://parlvu.parl.gc.ca/Harmony/en PowerBrowserPowerBrowserV2/20191212/671/32299.

[11] Canada, Parliment, and House of Commons. 43th parliament, question period for setting no.4 house of commons. https://parlvu.parl.gc.ca/Harmony/en/PowerBrowser/PowerBrowserV2/20191210/-1/32284.

[12] Canada, Parliment, and House of Commons. 43th parliament, setting no.1 house of commons. https://parlvu.parl.gc.ca/Harmony/en/PowerBrowser PowerBrowserV2/20191205/-1/32139.

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

[14] Xiao Chen, Yaan Li, Yuxing Li, Jing Yu, and Xiaohua Li. A novel probabilistic data association for target tracking in a cluttered environment. *Sensors*, 16(12):2180, 2016.

[15] Grigorios Chrysos, Epameinondas Antonakos, Patrick Snape, and Stefanos Zafeiriou. A comprehensive performance evaluation of deformable face tracking "in-the-wild". *International Journal of Computer Vision*, 126, 03 2016.

[16] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection via region-based fully convolutional networks. In *Advances in neural information processing systems*, pages 379–387, 2016.

[17] Caglayan Dicle, Octavia I Camps, and Mario Sznaier. The way they move: Tracking multiple targets with similar appearance. In *Proceedings of the IEEE international conference on computer vision*, pages 2304–2311, 2013.

[18] Dumitru Erhan, Christian Szegedy, Alexander Toshev, and Dragomir Anguelov. Scalable object detection using deep neural networks. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2155–2162, 2014.

[19] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[20] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[21] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[22] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[23] Yandong Guo, Lei Zhang, Yuxiao Hu, Xiaodong He, and Jianfeng Gao. Ms-celeb-1m: A dataset and benchmark for large-scale face recognition. In *European conference on computer vision*, pages 87–102. Springer, 2016.

[24] Seyed Hamid Rezatofighi, Anton Milan, Zhen Zhang, Qinfeng Shi, Anthony Dick, and Ian Reid. Joint probabilistic data association revisited. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3047–3055, 2015.

[25] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(9):1904–1916, 2015.

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

[28] Shaoming He, Hyo-Sang Shin, and Antonios Tsourdos. Joint probabilistic data association filter with unknown detection probability and clutter rate. *Sensors*, 18(1):269, 2018.

[29] Alexander Hermans, Lucas Beyer, and Bastian Leibe. In defense of the triplet loss for person re-identification. 03 2017.

[30] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. 2012.

[31] Jan Hosang, Rodrigo Benenson, and Bernt Schiele. Learning non-maximum suppression. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4507–4515, 2017.

[32] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications.

[33] Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, et al. Speed/accuracy trade-offs for modern convolutional object detectors. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7310–7311, 2017.

[34] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. 02 2015.

[35] Vidit Jain and Erik Learned-Miller. Fddb: A benchmark for face detection in unconstrained settings. Technical report, UMass Amherst technical report, 2010.

[36] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. Tracking-learning-detection. *IEEE transactions on pattern analysis and machine intelligence*, 34(7):1409–1422, 2011.

[37] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. Tracking-learning-detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1409–1422, July 2012.

[38] Laleh Rabiee Kenari and Mohammad Reza Arvan. Comparison of nearest neighbor and probabilistic data association methods for non-linear target tracking data association. In *2014 Second RSI/ISM International Conference on Robotics and Mechatronics (ICRoM)*, pages 047–052. IEEE, 2014.

[39] Chanho Kim, Fuxin Li, Arridhana Ciptadi, and James M Rehg. Multiple hypothesis tracking revisited. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4696–4704, 2015.

[40] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[41] Laura Leal-Taixé, Anton Milan, Ian Reid, Stefan Roth, and Konrad Schindler. MOTChallenge 2015: Towards a Benchmark for Multi-Target Tracking. *arXiv e-prints*, page arXiv:1504.01942, Apr 2015.

[42] Yann LeCun and Yoshua Bengio. *Convolutional Networks for Images, Speech, and Time Series*, page 255–258. MIT Press, Cambridge, MA, USA, 1998.

[43] Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

[44] Charay Lerdsudwichai and Mohamed Abdel-Mottaleb. Algorithm for multiple faces tracking. pages II – 777, 08 2003.

[45] Feifei Li. *https://cs231n.github.io/neural-networks-1/*. Feifei Li, May 2017.

[46] Haoxiang Li, Zhe Lin, Xiaohui Shen, Jonathan Brandt, and Gang Hua. A convolutional neural network cascade for face detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5325–5334, 2015.

[47] Pei Li, Loreto Prieto, Domingo Mery, and Patrick J Flynn. On low-resolution face recognition in the wild: Comparisons and new techniques. *IEEE Transactions on Information Forensics and Security*, 14(8):2000–2012, 2019.

[48] Hwasup Lim, Octavia Camps, Mario. Sznaier, and Vlad I. Morariu. Dynamic appearance modeling for human tracking. volume 1, pages 751– 757, 07 2006.

[49] Chung-Ching Lin and Ying Hung. A prior-less method for multi-face tracking in unconstrained videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 538–547, 2018.

[50] Yiming Lin, Shiyang Cheng, Jie Shen, and Maja Pantic. Mobiface: A novel dataset for mobile face tracking in the wild. In *2019 14th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2019)*, pages 1–8. IEEE, 2019.

[51] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[52] Wei Liu, Andrew Rabinovich, and Alexander C Berg. Parsenet: Looking wider to see better. 2015.

[53] Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song. Sphereface: Deep hypersphere embedding for face recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 212–220, 2017.

[54] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[55] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully Convolutional Networks for Semantic Segmentation. *arXiv e-prints*, page arXiv:1411.4038, Nov 2014.

[56] Nicolas Malasne, Fan Yang, and Michel Paindavoine. Real-time face tracking and recognition for video conferencing. *Proc SPIE*, 4474:357–365, 11 2001.

[57] Yoanna Martindez-Diaz, Luis S Luevano, Heydi Mendez-Vazquez, Miguel Nicolas-Diaz, Leonardo Chang, and Miguel Gonzalez-Mendoza. Shufflefacenet: a lightweight

face architecture for efficient and highly-accurate face recognition. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 0–0, 2019.

[58] Markus Mathias, Rodrigo Benenson, Marco Pedersoli, and Luc Van Gool. Face detection without bells and whistles. volume 8692, 09 2014.

[59] Anton Milan, Konrad Schindler, and Stefan Roth. Challenges of ground truth evaluation of multi-target tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 735–742, 2013.

[60] Shruti Nagpal, Maneet Singh, Richa Singh, Mayank Vatsa, and Nalini Ratha. Deep learning for face recognition: Pride or prejudiced? *arXiv preprint arXiv:1904.01219*, 2019.

[61] Aaron Nech and Ira Kemelmacher-Shlizerman. Level playing field for million scale face recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7044–7053, 2017.

[62] Omkar M Parkhi, Andrea Vedaldi, and Andrew Zisserman. Deep face recognition. 2015.

[63] Federico Pernici. Facehugger: The alien tracker applied to faces. In *European Conference on Computer Vision*, pages 597–601. Springer, 2012.

[64] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[65] Donald Reid. An algorithm for tracking multiple targets. *IEEE transactions on Automatic Control*, 24(6):843–854, 1979.

[66] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[67] Oren Rippel, Manohar Paluri, Piotr Dollar, and Lubomir Bourdev. Metric learning with adaptive density discrimination. *arXiv preprint arXiv:1511.05939*, 2015.

[68] Pierre Sermanet, David Eigen, Xiang Zhang, Michael Mathieu, Rob Fergus, and Yann LeCun. OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks. *arXiv e-prints*, page arXiv:1312.6229, Dec 2013.

[69] Mohd Razif Shamsuddin, Shuzlina Rahman, and Azlinah Mohamed. *Exploratory Analysis of MNIST Handwritten Digit for Machine Learning Modelling: 4th International Conference, SCDS 2018, Bangkok, Thailand, August 15-16, 2018, Proceedings*, pages 134–145. 01 2019.

[70] Andrew Jason Shepley. Deep learning for face recognition: A critical analysis. *arXiv preprint arXiv:1907.12739*, 2019.

[71] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[72] Yi Sun, Xiaogang Wang, and Xiaoou Tang. Deep learning face representation from predicting 10,000 classes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1891–1898, 2014.

[73] Kah-Kay Sung and Tomaso Poggio. Example-based learning for view-based human face detection. *IEEE Transactions on pattern analysis and machine intelligence*, 20(1):39–51, 1998.

[74] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[75] Arsene Fansi Tchango, Vincent Thomas, Olivier Buffet, Alain Dutech, and Fabien Flacher. Tracking multiple interacting targets using a joint probabilistic data association filter. In *17th International Conference on Information Fusion (FUSION)*, pages 1–8. IEEE, 2014.

[76] Daniel Sáez Trigueros, Li Meng, and Margaret Hartnett. Face recognition: from traditional to deep learning methods. *arXiv preprint arXiv:1811.00116*, 2018.

[77] Régis Vaillant, Christophe Monrocq, and Yann Le Cun. Original approach for the localisation of objects in images. *IEE Proceedings - Vision, Image and Signal Processing*, 141(4):245–250, 1994.

[78] Paul Viola and Michael Jones. Robust real-time face detection. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, volume 2, pages 747–747, 2001.

[79] Derui Wang, Chaoran Li, Sheng Wen, Surya Nepal, and Yang Xiang. Daedalus: Breaking non-maximum suppression in object detection via adversarial examples. *arXiv preprint arXiv:1902.02067*, 2019.

[80] Mei Wang and Weihong Deng. Deep face recognition: A survey. *arXiv preprint arXiv:1804.06655*.

[81] Kilian Q Weinberger and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10(Feb):207–244, 2009.

[82] Nicolai Wojke and Alex Bewley. Deep cosine metric learning for person re-identification. In *2018 IEEE winter conference on applications of computer vision (WACV)*, pages 748–756. IEEE, 2018.

[83] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. In *2017 IEEE international conference on image processing (ICIP)*, pages 3645–3649. IEEE, 2017.

[84] Lior Wolf, Tal Hassner, and Itay Maoz. Face recognition in unconstrained videos with matched background similarity. In *CVPR 2011*, pages 529–534. IEEE, 2011.

[85] Yu Xiang, Alexandre Alahi, and Silvio Savarese. Learning to track: Online multi-object tracking by decision making. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 4705–4713, 2015.

[86] Hanxuan Yang, Ling Shao, Feng Zheng, Liang Wang, and Zhan Song. Recent advances and trends in visual tracking: A review. *Neurocomputing*, 74:3823–3831, 11 2011.

[87] Shuo Yang, Ping Luo, Chen-Change Loy, and Xiaoou Tang. Wider face: A face detection benchmark. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5525–5533, 2016.

[88] Dong Yi, Zhen Lei, Shengcai Liao, and Stan Z Li. Learning face representation from scratch. *arXiv preprint arXiv:1411.7923*, 2014.

[89] Ju Hong Yoon, Ming-Hsuan Yang, Jongwoo Lim, and Kuk-Jin Yoon. Bayesian multi-object tracking using motion context from multiple objects. In *2015 IEEE Winter Conference on Applications of Computer Vision*, pages 33–40. IEEE, 2015.

[90] Liping Yuan, Zhiyi Qu, Yufeng Zhao, Hongshuai Zhang, and Qing Nian. A convolutional neural network based on tensorflow for face recognition. In *2017 IEEE*

2nd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), pages 525–529. IEEE, 2017.

[91] Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, and Yu Qiao. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters*, 23(10):1499–1503, 2016.

[92] Kunlei Zhang, Elaheh Barati, Elaheh Rashedi, and Xue-Wen Chen. Long-term face tracking in the wild using deep learning. 08 2016.

[93] Shun Zhang, Yihong Gong, Jia-Bin Huang, Jongwoo Lim, Jinjun Wang, Narendra Ahuja, and Ming-Hsuan Yang. Tracking persons-of-interest via adaptive discriminative features. In *European conference on computer vision*, pages 415–433. Springer, 2016.

[94] Wen-Yi Zhao, Rama Chellappa, P. Jonathon Phillips, and Azriel Rosenfeld. Face recognition: A literature survey. *ACM Comput. Surv.*, 35:399–458, 12 2003.

[95] Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu, and Xindong Wu. Object detection with deep learning: A review. *IEEE transactions on neural networks and learning systems*, 30(11):3212–3232, 2019.

[96] Jingxiao Zheng, Rajeev Ranjan, Ching-Hui Chen, Jun-Cheng Chen, Carlos D Castillo, and Rama Chellappa. An automatic system for unconstrained video-based face recognition. *arXiv preprint arXiv:1812.04058*, 2018.