Adaptation of Deep Siamese Neural Networks for Video Face Recognition

by

Hugo LEMOINE ST-ANDRE

THESIS PRESENTED TO ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
IN PARTIAL FULFILLMENT FOR A MASTER'S DEGREE
WITH THESIS IN SYSTEM ENGINEERING
M.A.Sc.

MONTREAL, OCTOBER 5, 2021

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

BOARD OF EXAMINERS

THIS THESIS HAS BEEN EVALUATED

BY THE FOLLOWING BOARD OF EXAMINERS

Professor Eric Granger, memorandum supervisor
System Engineering Department, École de technologie supérieure

Researcher Mohamed Dahmane, co-supervisor
Vision and Imaging Department, Centre de Recherche Informatique de Montréal

Professor Matt Toews, member of the jury
System Engineering Department, École de technologie supérieure

Professor Georges Kaddoum, member of the jury
Electrical Engineering Department, École de technologie supérieure

THIS THESIS WAS PRESENTED AND DEFENDED

IN THE PRESENCE OF A BOARD OF EXAMINERS AND THE PUBLIC

ON AUGUST 31, 2021

AT ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

ACKNOWLEDGEMENTS

I would like to express my gratitude to professor Eric Granger and my co-director Mohamed Dahmane for the opportunity and guidance in computer vision and deep learning research. I have learned much and it will help me with my future career.

I would like also to thank Ghenadie Odobescu and for supporting my research. With this support, I could focus my work around real world application, which will help me a lot for applying complex deep learning solution to product.

Special thanks to George Ekladious to push my research with new ideas that allowed me to learn a lot. Your knowledge and patience is formidable and I am truly grateful for that. I definitely had my best time working with you during my research.

Thank you very much to my colleagues from LIVIA who help me to understand and to advance my research. Big thanks to Jerome Rony, who gave me a hand when I needed the most for technical problems and took the time to help me. Also, thank you to Madhu Kiran and Imtiaz Masud Ziko for the help and tips.

Finally, thank you to my family and friends who supported me during the writing of my thesis. Thank you to my wife and my mother for the encouragement and many, many thanks to Natasha Lepore for the guidance and wisdom. I could not do it without you.

Adaptation de réseaux de neurones profonds siamois pour la reconnaisance de visage dans des vidéo

Hugo LEMOINE ST-ANDRE

RÉSUMÉ

La reconnaissance faciale pour des images statiques de visage a été très explorée et généralement avec succès, mais des images vidéo de visages capturés dans des environnements non contraints pose des défis plus difficiles puisque les images souffrent plus de variation de pose, de flou, d'illumination, de basse résolution et de basse qualité. Dans cette thèse, nous adressons la reconnaissance faciale pour des applications vidéo. Premièrement, nous explorons la re-identification faciale pour des applications vidéos en essayant de comparer des paires de visages pour identifier une personne dans une base de données en utilisant un réseau Siamois profond. Après, nous explorons la description vidéo en essayant de capturer la distribution des identités d'un film en utilisant un réseau Siamois avec une technique de catégorisation. Pour adresser ce problème, d'autres recherches utilisent une quantité large de données annotées pour entraîner leurs modèles, ce qui est problématique puisque l'annotation est couteuse en temps et ressources. L'objectif est d'adapter un réseau Siamois profond entrainé avec des bases de données publiques composées d'images statiques de visage à un domaine vidéo non contraint d'une manière non supervisée, ce qui enlèverait la nécessité d'annoter manuellement. À cette fin, nous utilisons triplet-loss pour apprendre et adapter une représentation faciale discriminante d'une manière pratique pour des applications réelles.

Des recherches récentes en vidéo surveillance utilisent de l'adaptation supervisé pour réduire l'écart entre les images statiques et vidéos. D'autres recherches utilisent de l'adaptation de domaine faiblement-supervisé ou non supervisé, mais peu utilisent des réseaux Siamois profonds. Ceux-ci requièrent que le domaine ciblé soit un problème avec un nombre de classes prédéterminé ou d'avoir une grande quantité de données non annotées ce qui n'est pas pratique. Dans cette thèse, nous introduisons l'apprentissage par duo de triplet qui est une variante de l'apprentissage par triplet. Nous utilisons simultanément des triplets du domaine source et vidéo pour adapter les représentations robustes d'image statiques à une source vidéo nouvellement installée en utilisant peu de données non annotées. La méthodologie est validée avec la base de donnée COX-S2V où nous obtenons un gain en précision de classification de 3% à 7%.

En ce qui concerne la description vidéo, nous utilisons des réseaux Siamois profonds avec l'information de tracklet pour grouper les visages de même identité. Avec un tel outil, il serait possible de décrire des visages de n'importe quel vidéo automatiquement. À cette fin, nous adaptons un CNN robuste de visage statique à un film en utilisant les visages non annotés de ce même film. En utilisant l'information spatio-temporelle de la vidéo, il est possible de produire des pairs de visage pour l'apprentissage par triplet. Avec cela, nous essayons l'apprentissage auto-supervisé avec un réseau Siamois profond, utilisant le premier épisode de la série télévisée The Big Bang Theory, pour apprendre des caractéristiques robustes des visages présents dans le film. Nous démontrons que l'apprentissage auto-supervisé peut améliorer le

score de regroupement (mesure V) de 15%. De plus, avec un nombre suffisant de données, les trackets peuvent être utilisés comme représentation simple pour faire du regroupement plus rapidement et précisément.

Mots-clés: reconnaissance faciale, apprentissage profond, réseau Siamois, vidéo surveillance, vidéo description.

Adaptation of Deep Siamese Neural Networks for Video Face Recognition

Hugo LEMOINE ST-ANDRE

ABSTRACT

Face recognition for static face images has been well explored and is generally very successful, but video face images taken in unconstrained environments pose more difficult challenges as the image samples suffer from more issues such as pose variation, blur, illumination variation, lower resolution and lower quality. This thesis, addresses face recognition for video-based applications. First, we explore face re-identification for video surveillance applications, attempting pairwise face matching to identify a person in a database using a deep Siamese network. Next, we explore video description, attempting to capture the distribution of the identity samples from a movie using the same Siamese network with clustering techniques. To address this problem, other researchers have labeled a large amount of data in order to enhance their model, which is problematic as it is time- and resource-intensive. The objective is to a adapt deep Siamese network trained on public datasets of static face images to the unconstrained video domain in an unsupervised manner, removing the need to label data manually. To this end, we use triplet loss to learn and adapt discriminative face features in a practical manner for real-world video applications.

Recent work in video surveillance has used supervised adaptation to close the domain gap between static images and videos. Other researchers have used weakly-supervised or unsupervised domain adaptation, but there are very few works based on a deep Siamese network. These require the target domain to be either a closed-set problem or have a very large amount of unlabeled data, both of which impractical. In this thesis, we introduce an unsupervised domain adaptation named Dual-Triplet learning which is a variant of triplet learning. It simultaneously uses triplets from source and target domains to adapt robust static representation to newly installed video sources using only a few unlabeled samples. The methodology is validated with the COX-S2V dataset whith which we are able to get 3% to 7% gain in classification accuracy.

In regards to video description, we intend to use a deep Siamese network with tracklet information to group face samples of the same identity. With such a tool, it will be possible to describe faces on any video automatically. To this end, robust static face CNN backbones are adapted to a movie using unlabeled data from the movie itself. By using spatio-temporal information (tracklets) of video samples, it is possible to produce positive and negative pairs for triplet loss training. With this, we attempt self-supervised learning with a deep Siamese network, using the first episode of the television series The Big Bang Theory, to learn robust and discriminative features of face samples from the movie. We show that self-supervised learning can enhance the clustering V measure by 15%. We also show that with a sufficient number of samples, the tracklets can be used as a single representation to perform faster and more accurate clustering.

Keywords: face recognition, deep learning, Siamese network, video surveillance, video description

TABLE OF CONTENTS

Page

INTRODUCTION 1

CHAPTER 1 RELATED WORKS 5
1.1 Introduction 5
1.2 Deep Learning Models 5
1.2.1 Convolutional Neural Networks 5
1.2.2 Siamese Networks 10
1.3 Face Recognition 14
1.3.1 Detection 14
1.3.2 Matching 15
1.4 Unsupervised Domain Adaptation 17

CHAPTER 2 EXPERIMENTAL METHODOLOGY 19
2.1 Datasets 19
2.1.1 VGGface2 19
2.1.2 LFW 19
2.1.3 COX-S2V 20
2.1.4 The Big Bang Theory TV series 23
2.2 Performance Measures 23
2.2.1 Classification Measures 24
2.2.2 Clustering Measures 25
2.2.3 Complexity 28

CHAPTER 3 FACE RECOGNITION WITH SIAMESE NETWORKS FOR VIDEO SURVEILLANCE 31
3.1 Introduction 31
3.2 Face Recognition for Video Surveillance 33
3.2.1 Detection and Tracking 34
3.2.2 Pair-wise Matching 35
3.3 Supervised Domain Adaptation of Video Face Representations 37
3.4 Unsupervised Domain Adaptation of Video Face Representations using Dual-Triplet Metric Learning 38
3.4.1 Dual-Triplet Metric Learning 39
3.4.2 Dual-Triplet Loss 41
3.4.3 Mutual Supervision 42
3.4.4 Experimental Methodology 44
3.4.5 Experimental Results and Discussion 46
3.5 Conclusion 50

CHAPTER 4 A SIAMESE NETWORK FOR VIDEO DESCRIPTION 51

4.1 Introduction 51
4.2 Face Recognition for Video Description 52
4.2.1 Segmentation 52
4.2.2 Feature Extractor 53
4.2.3 Clustering 53
4.3 Video Description with Pre-Trained CNN Backbone 55
4.3.1 Experiment on Small Video Segment 56
4.3.2 Experiment on Large Video Segment 58
4.4 Face Clustering Experiment with Domain Adaptation Based on Tracklets 65
4.5 Conclusion 69

CONCLUSION AND RECOMMENDATIONS 71

BIBLIOGRAPHY 73

LIST OF TABLES

Page

Table 3.1 Base FE LFW evaluation metics. 37

Table 3.2 Base feature extractor on COX S2V datasets. 38

Table 3.3 Results on the COX Face Recognition dataset using the proposed DTML method with mutual-Supervision. 45

Table 3.4 Comparison of model complexity for video surveillance. 49

Table 4.1 Clustering results on the first 500 frames of BBT episode 1. 56

Table 4.2 Clustering results on BBT episode 1 at frame level. 58

Table 4.3 Clustering results on BBT episode 1 at frame level per number of clusters. 59

Table 4.4 Clustering results on BBT episode 1 at track level. 60

Table 4.5 Clustering results on BBT episode 1 at track level per number of clusters. 63

Table 4.6 HAC results on BBT episode 1 at frame level with tracklets fine-tuned. 67

Table 4.7 HAC results on BBT episode 1 at track level with tracklets fine-tuned. 67

Table 4.8 HAC with 5 clusters confusion matrix on BBT episode 1 at track level with tracklets fine-tuned. 68

Table 4.9 HAC clustering V measure on BBT episode 1 comparison. 70

LIST OF FIGURES

Page

Figure 1.1 Example of convolutional neural network (CNN). 6

Figure 1.2 Example of convolution. 7

Figure 1.3 ResNet residual block 9

Figure 1.4 Siamese neural network. 11

Figure 1.5 Triplet difficulty distribution. 13

Figure 2.1 VGGface2 samples 20

Figure 2.2 COX-S2V capture path 21

Figure 2.3 COX-S2V still sample. 21

Figure 2.4 COX-S2V video samples. 22
(a) CAM1/video4 22
(b) CAM2/video3 22
(c) CAM3/video2 22
(d) CAM3/video1 22

Figure 3.1 Face recognition system for video surveillance. 34

Figure 3.2 FAce STructured Detection and Tracking 35

Figure 3.3 Face identification with a Siamese network. 36

Figure 3.4 Base FE training metrics. 36
(a) Triplet total loss. 36
(b) Triplet mining. 36

Figure 3.5 Base FE evaluation metrics on LFW. 37
(a) LFW validation accuracy during training. 37
(b) LFW distance pairs distribution. 37

Figure 3.6 COX-S2V distance distribution with base feature extractor. 39
(a) video 1 39
(b) video 2 39
(c) video 3 39
(d) video 4 39

Figure 3.7 Dual-triplet metric learning for domain adaptation 40

Figure 3.8 Illustration of the mutual supervision learning 43
(a) Source: represented by the initial source embedding 43
(b) Target: represented by the initial source embedding 43
(c) Source: represented by a shared embedding through DTML..... 43
(d) Target: represented by a shared embedding through DTML 43

Figure 3.9 AUC performance on COX dataset camera 1 to camera 3 47

Figure 3.10 AUC performance on COX dataset camera 3 to camera 2 47

Figure 3.11 COX dataset: TSNE representation for the first ten people in the testing set..... 48
(a) Before domain adaptation 48
(b) After domain adaptation. 48

Figure 4.1 Face recognition system for video description 52

Figure 4.2 Example of hierarchical agglomerative clustering 55
(a) Samples in Euclidean space 55
(b) Hierarchical link between samples..... 55

Figure 4.3 TSNE projections of BBT episode 1 reduced..... 57
(a) Ground truth: 2 clusters 57
(b) HAC: 2 clusters 57
(c) Affinity propagations: 3 clusters 57
(d) Kmeans: 2 clusters..... 57
(e) Kmodes: 3 clusters 57
(f) Spectral: 3 clusters..... 57

Figure 4.4 Frame level TSNE projections of BBT episode 1 with predicted clustering. 59
(a) Ground truth: 7 clusters 59
(b) HAC: 2 clusters 59
(c) Kmeans: 2 clusters..... 59
(d) Spectral: 2 clusters..... 59

Figure 4.5 Frame level silouhette score per k clusters. 60

Figure 4.6 Frame level TSNE projections of BBT episode 1 with selected number of clusters..... 61
(a) Ground truth: 7 clusters 61
(b) HAC: 4 clusters 61
(c) HAC: 7 clusters 61

(d) Kmeans: 4 clusters 61
(e) Kmeans: 7 clusters 61

Figure 4.7 TSNE projection of BBT episode 1 at track level. 62
(a) Ground truth: 7 clusters 62
(b) HAC: 2 clusters 62
(c) Affinity propagations: 4 clusters 62
(d) Kmeans: 2 clusters 62
(e) Kmodes: 2 clusters 62
(f) Spectral: 2 clusters 62

Figure 4.8 Track level silouhette score per k clusters. 63

Figure 4.9 Track level TSNE projections of BBT episode 1 with selected number of clusters. 64
(a) Ground truth: 7 clusters 64
(b) HAC: 4 clusters 64
(c) HAC: 7 clusters 64
(d) Kmeans: 4 clusters 64
(e) Kmeans: 7 clusters 64

Figure 4.10 Frame level TSNE projections of BBT episode 1 with tracklets fine-tuned. 66
(a) Ground truth: 7 clusters 66
(b) HAC: 4 clusters 66
(c) HAC: 5 clusters 66
(d) HAC: 7 clusters 66

Figure 4.11 Track level TSNE projections of BBT episode 1 with tracklets fine-tuned. 68
(a) Ground truth: 7 clusters 68
(b) HAC: 4 clusters 68
(c) HAC: 5 clusters 68
(d) HAC: 7 clusters 68

LIST OF ABREVIATIONS

CNN	Convolutional Neural Network
DTML	Dual-Triplet Metric Loss
FE	Feature Extractor
FR	Face Recognition
ROI	Region of Interest
S2S	Still to Still
S2V	Still to Video
UDA	Unsupervised Domain Adaptation
V2V	Video to Video

INTRODUCTION

One of the most well known applications in pattern recognition is facial recognition. Face recognition has become very popular because of its potential to identify human faces discretely and effectively. It can be applied, for example, in video surveillance, biometric identification, and in video description. Currently, state-of-the-art face recognition techniques are based on deep learning architectures.

The pattern recognition domain took a huge leap forward in 2012 with the successful application of a convolutional neural network (CNN), as reported by the first CNN AlexNet (Krizhevsky, Sutskever & Hinton, 2012). A CNN is a deep learning architecture specifically designed to learn patterns from image data. It usually consists of millions of parameters that are adjusted through gradient descent (SGD).

However, even with the arrival of deep CNN, facial recognition remains a challenge. To perform facial recognition successfully, many factors must be considered such as illumination, occlusion, blur and pose (Jain & Li, 2011). Also, we must consider that in the case of facial recognition, the problem is an open-set classification problem for which there are an indefinite number of identities alias classes. Using representation learning with Siamese networks allow CNN to operate on open-set problems by learning discriminative features of faces (Bengio, Courville & Vincent, 2013).

Face can be captured in a frontal high-quality picture of a still subject or from a video where the subject is moving. Subjects captured from video suffer from the effects of blur and pose variations. Therfore, we can formulate three cases of comparison: still-to-still (S2S), still-to-video (S2V), and video-to-video (V2V). Still-to-still works by using a high-quality still region of interest (ROI) as a reference. This is the easiest method, because the ROI is captured in a controlled environment and the image is of adequate quality. With still-to-video, the reference ROIs are taken from still images and the testing ROIs are taken from a video stream. Here, the difference

in capture conditions can pose a challenge. Finally, video-to-video occurs when a video-captured ROI is compared to a video-captured reference.

For the kind of video surveillance that seeks to retrieve a subject, the facial-recognition system operates in a S2V application. The researched subjects are registered with high quality still images. Then, a surveillance camera can capture subjects to compare them with the registered subjects. The system will notify the user if there is a match between the subjects.

Problem statement

In this thesis, we will address two challenges of using facial recognition for video surveillance and video description applications. In regards to video surveillance, we attempt to identify subjects in video samples from a database of still face images. In regards to video description, we will attempt automatic regrouping of identities from videos for name-tagging purposes.

A face recognition system has three main modules. It must perform face segmentation (or detection), face representation, and similarity comparison. The segmentation process has been studied by many groups (Hsu, 2002) (Yang, Luo, Loy & Tang, 2016) (Jain & Learned-Miller, 2010). CNN-based face detection systems have had great success using tracking algorithms to capture faces in various scenes and to produce trajectories. These techniques have become so efficient, they can capture faces with various poses and from images of different quality and illumination. These variations pose a key problem for similarity comparison as it must be able to produce discriminative representation from challenging ROI.

The face representations are trained using a massive database that is typically made up of images of known individuals taken from the internet. However, these annotated images may not have the same camera-capturing conditions. Having annotated samples from a scene, it would be possible to enhance the representation for better performance with that specific scene using

supervised domain adaptation. This would close the domain gap between the source data and the camera-captured conditions, but comes at the price of greater resource and time requirements.

For autonomous video description, regrouping the face representations of a single character can be difficult. As face segmentation is very robust, it captures the face of an individual in various conditions. In this particular case, the pose of the individual is a critical element. Two trajectories of the same character with different poses can create two distinctive groups in the feature space.

Challenges

The challenges for facial recognition that I will address in this manuscript are as follows:

- **The domain shift between static and video face images**: State-of-the-art face representations are deep learning models that require many annotated samples to produce discriminative representations of identities. As face segmentation becomes more powerful, we include more small image samples or samples with more variations in pose and occlusion. It is then more challenging for the feature extractor to produce those representations as it must learn these new complexities. Generally, the video domain in which we perform facial recognition does not have annotated samples. As manually annotating samples is time and resource-intensive, the challenge is to find a way to perform domain adaption in a unsupervised or self-supervised manner using only a few unlabeled samples.

- **Face clustering**: In movies or TV episodes, face segmentation can capture many identities in various conditions. Regrouping those faces in the feature space is difficult because the pose, illumination and image quality cannot produce distinctive clusters. Also, it is possible to have heavily imbalanced data, making it difficult for a deep learning method with a clustering algorithm to differentiate clearly between an identity with few occurrences and an identity with many occurrences.

Research Objectives and Contributions

The research objective is to develop deep learning models based on CNN for facial recognition. Specifically, the objectives are to perform facial recognition for video surveillance with a Siamese network; to propose a simple methodology for unsupervised or self-supervised domain adaptation for video surveillance and video description.

The contributions are as follows:

- In chapter 3, we explain the representation learning process of our base face representation and we apply unsupervised domain adaptation (UDA) for video surveillance on COX-S2V dataset. Our UDA is based on our proposed dual-triplet metric learning and mutual supervised sample mining. This work was presented in (Ekladious, Lemoine, Granger, Kamali & Moudache, 2020).
- In chapter 4, we use self-supervised DA to perform video description of the first episode of the television series The Big Bang Theory. This is achieved by exploiting the spatio-temporal information from face trajectories called tracklets.

CHAPTER 1

RELATED WORKS

1.1 Introduction

In this chapter, we will review the literature of deep neural network techniques for face recognition tasks. First, we will review convolutional neural networks (CNNs) and their components which represent the state-of-the-art techniques to solve pattern recognition problems. Then, we will summarize the application of CNNs in the face recognition field. Finally, we will review some techniques to perform domain adaption for face recognition successfully across different video domains in an unsupervised manner.

1.2 Deep Learning Models

1.2.1 Convolutional Neural Networks

Since the success of a convolutional neural network (CNN) in 2012 with AlexNet (Krizhevsky *et al.*, 2012) in the ImageNet competition, deep learning techniques have become widely used for solving computer vision problems. CNNs have become deeper and deeper to achieve better accuracy in image classification. Some of the most popular CNN architectures were the AlexNet, VGG network (Parkhi, Vedaldi & Zisserman, 2015), Inception (Szegedy, Liu, Jia, Sermanet, Reed, Anguelov, Erhan, Vanhoucke & Rabinovich, 2015) and ResNet (He, Zhang, Ren & Sun, 2016). These were achieved due to the availability of greater computational resources.

A CNN is a deep learning architecture that uses images as input to produce discriminative features learned from a large dataset. These features can then be used for classification or regression tasks. CNN architectures are used for image recognition due to their capacity to maintain spatial information across the network.

In this section, we will cover the basics of CNN operations and review some popular architectures and their characteristics.

Components

The general components of the CNNs reviewed in this section are the convolution layer, the activation function, pooling, and the fully connected layer. Figure 1.1 shows how these components are assembled together to form the network.

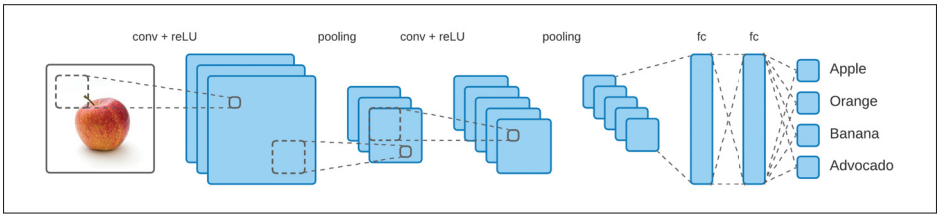


Figure 1.1 Example of convolutional neural network (CNN).

Convolution Layer. The convolution layer is made up of learned kernels that are passed over the feature map or the input image. Each convolution layers features multiple kernels that learn different small patterns. The kernel is a 2D matrix of numbers of n by n size, and the number of kernels can vary, going as high as 4,096 per convolution layer for certain CNN architectures. The convolution process entails multiplying each n by n section of the image by each kernel. Given an output feature map of the size w x h with c kernels, it gives a feature map with shape image w x h x c. To gradually reduce the size of the feature map, one could use pooling techniques.

Activation Function. At the end of the convolution operation, we apply an activation function. The activation function allows the introduction of non-linearity into the CNN. Indeed, pattern recognition problems are generally nonlinear, while the operations used in CNNs such as addition and multiplication are linear. Without introduced non-linearity, all of the layers would be summarized into a simple linear function. The ReLU activation function is generally used because it works well and is fast to compute. For binary classification, we use a sigmoid function where the output value is between 0 and 1, which reflects confidence probability.

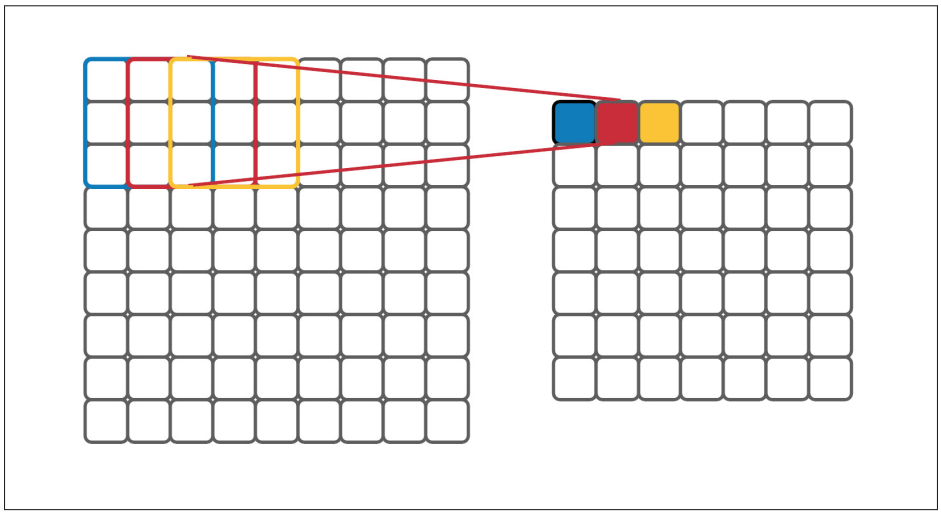


Figure 1.2 Example of convolution.

Pooling. In a CNN architecture, pooling operations reduce the size of a feature map. This reduction concentrates the features and makes the classification process easier, as the space becomes less complex. However, reducing the space with pooling leads to information loss. Generally, architectures use max pooling or average pooling, which reduce sections of the feature map with the maximum value or average value, respectively, within that section.

Fully Connected Layer. Fully connected layers are the classic neural networks in which all neurons in one layer connect to all neurons of the next layer. They are known to be inefficient when performing visual tasks, compared to convolution networks. Usually, they are placed at the end of a CNN network to perform classification or regression tasks. Their objective in a CNN is to flatten the results of pooling or convolution, to produce labels.

Architectures

We review some popular CNN architectures that are used in state-of-the-art techniques. These include AlexNet, VGG, ResNet and Inception architectures.

AlexNet. The AlexNet deep neural network was the first CNN to achieved state-of-the-art results in the ImageNet LSVRC-2012 competition (Krizhevsky *et al.*, 2012). The availability of greater

computational resources provided the opportunity to train such an ambitious network. The network parameters calculation was split among multiple GPUs to train more quickly.

VGG. The VGG neural network is one successor of the well-known AlexNet (Parkhi *et al.*, 2015). Compared to other networks of the time, the VGG focused more on the CNN depth. There are two variants of depth, consisting of 16 and 19 convolution layers, followed by 3 fully connected layers.

The success of the VGG network comes from the use of small 3x3 receptive fields with a stride of 1, compared to AlexNet with 11x11 receptive fields and a stride of 4. Stacking multiple convolution layers of 3x3 filter can give the same coverage as convolution with larger filters, but with less parameters. This allowed to make the network deeper since each layer have less parameters to train.

ResNet. As researchers sought to improve CNN performance by increasing the number of layers, they encountered the problem of the vanishing gradient. As the gradient is back-propagated from one layer to another, it tends to gradually become infinitely smaller. As the network goes deeper it may be harder to train leading to decreased performance.

The author thus proposed the ResNet architecture, which tackled the problem of the vanishing gradient by introducing residual blocks (He *et al.*, 2016). The residual block provides an easier way for the back-propagated gradient to flow through the whole network. It accomplishes this by learning the residual F(x). The residual block is shown in Figure 1.3.

There are multiple possible configurations of the ResNet architecture. One can choose a network depth of 18, 34, 50 or 101. The authors tried greater depths but these did not provide high gains in performance compared to ResNet-101 and in some cases it gave worse results.

Loss functions

Here, we will review two common loss functions for classification: the cross-entropy and Hinge loss functions.

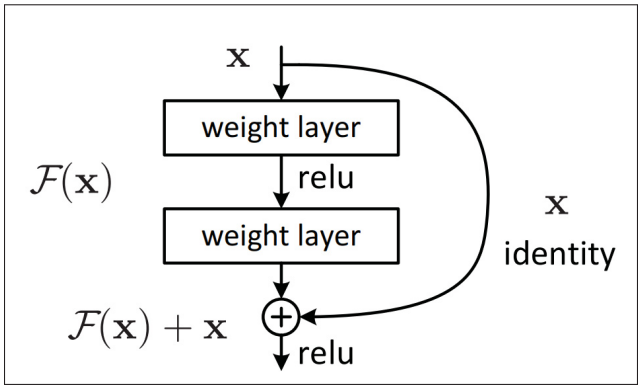


Figure 1.3 ResNet residual block (He *et al.*, 2016).

Cross-entropy loss function. The cross-entropy, or log loss function, is the measure of the performance of a classification prediction for which the value is between 0 and 1. This loss function is known to give a much higher loss value when the prediction score is different from the actual label and to give a lower loss value when the prediction is close to the actual label. A log loss value of 0 indicates a perfect prediction. The cross-entropy loss for multi-classification is calculated as follows:

$$L_{cross-entropy} = -\sum_{c=1}^{M} y_{o,c} log\left(p_{o,c}\right) \tag{1.1}$$

where M is the number of classes, y a binary indicator of 1 or 0 for if the label c is the true label of the observation o and p is the prediction score of the label c for the observation o.

Hinge loss function. The Hinge loss function is most frequently used with support vector machines, but is also used to train classifiers. Essentially, it is a simple linear function; the more the prediction value differs from the objective, the greater the loss value will be, linearly. If the prediction value meets the objective, the value will be 0. The equation is as follows:

$$L_{hinge} = \sum_{j \neq y_i} \max\left(0, 1 - y_i \cdot s_i\right) \tag{1.2}$$

where y is the label value -1 or 1, j is the image prediction, and s is the raw score of the image i.

1.2.2 Siamese Networks

In this section, we firstly describe the general architecture of a Siamese network. Then, we describe some loss function to train object representation for this architecture.

Architectures

Deep Siamese neural networks, are neural networks that can produce discriminative representation from an input for pairwise matching based on a distance metric. As the two images pass through the same neural network, their feature vectors can be compared in order to measure the similarity between those two images, as shown at Figure 1.4. Siamese networks are used, for example, for handwritten checks and face recognition tasks. The main advantage of using a Siamese network is to be able to classify an unlimited number of identities since the network is not using a classification layer. We can say that the Siamese architecture resembles to a one-shot classifier since we can retrieve an identity never seen during the training with only one reference sample from that identity.

For a Siamese network to perform, it requires a neural network able to produce discriminative spatial features for the specific objects to be matched. There are many methods to train a neural network to achieve this objective with metric learning such as contrastive loss and triplet loss.

Generally, a Siamese network is used to retrieve the identity of one or multiple objects across a set of input images. The images are processed by the neural network and the resulting representation vectors are stored in a database ready to be compared to newly arrived representation vectors.

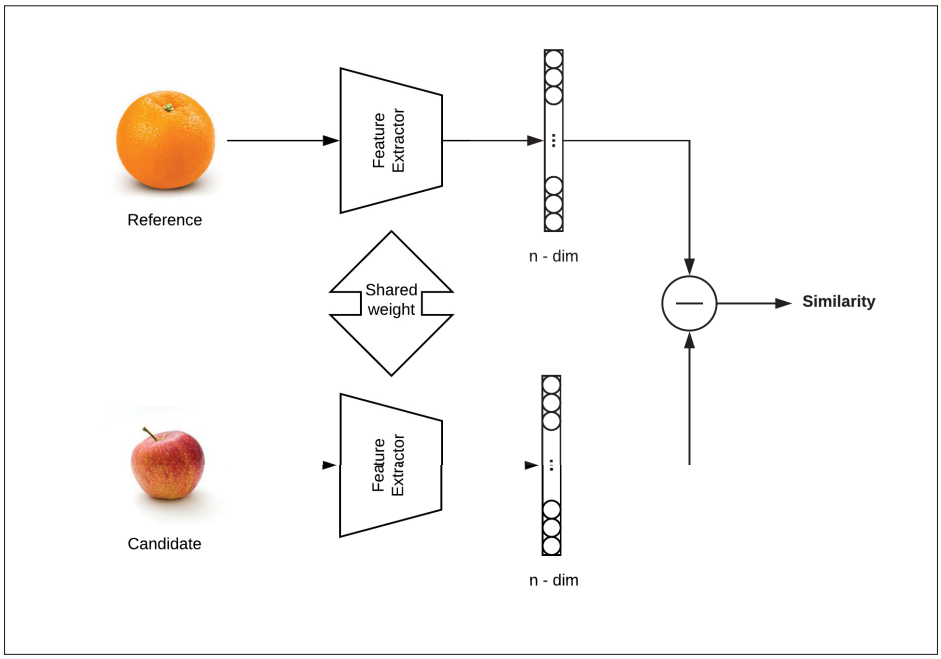


Figure 1.4 Siamese neural network.

Loss Functions for Metric Learning

In this subsection, we will review some loss functions that learn discriminant representations. Contrastive loss, triplet loss and Center loss were commonly use as Euclidean distance-based loss functions.

Contrastive Loss. Contrastive loss was first introduced in (Hadsell, Chopra & LeCun, 2006). It requires an ensemble of positive data pairs (same class) and negative pairs (different class). The contrastive loss is defined as follows:

$$L_{contrastive} = y_{ij} \max\left(\left\|f\left(x_i\right) - f\left(x_j\right)\right\|^2 - \epsilon^+\right) + \left(1 - y_{ij}\right) \max\left(\epsilon^- - \left\|f\left(x_i\right) - f\left(x_j\right)\right\|^2\right) \quad (1.3)$$

The term y_{ij} is equal to 1 if the image pair is positive and is equal to 0 if the image pair is negative. The terms x_i and x_j are the image representations that come in pairs. The ϵ^+ and ϵ^- terms control the margin of the positive pairs and negative pairs, respectively.

Triplet Loss. To enhance the performance of our face recognition algorithm, we attempt to refine the face feature extractor so that it can produce discriminative face representations for our specific cases. It is necessary to refine the model as we experiment, so that the base model does not provide optimal performance in domains other than the source domain. Our feature extractor is trained using a very large publicly available dataset VGGface2 (Q. Cao, Shen, Xie, Parkhi & Zisserman, 2018) that permits us to achieve good generalization of face characteristics. The training methodology is triplet loss where the model attempts to learn a margin between classes.

Triplet loss is a metric learning technique that learns a margin between inter-classes (Schroff, Kalenichenko & Philbin, 2015b). It learns discriminative embedding where the similarity can be measured with the distances between them. In the Euclidean space, the triplet loss is defined as follows:

$$L_{triplet} = \max\left(\|f(a) - f(p)\|^2 - \|f(a) - f(n)\|^2 + \alpha, 0\right) \tag{1.4}$$

Here, a is an anchor, p is a positive sample to the anchor, n is a negative sample to the anchor and f is the feature extractor. Once the loss function approaches to zero, the positive samples should be near to each other in the Euclidean space and the negative samples should be separated by the defined margin α.

The cost function to minimize is then expressed by the following equation where M is the number of trainable triplets in the training set:

$$\sum_{i=1}^{M} L_{triplet}(a_i, p_i, n_i) \tag{1.5}$$

Selection of the triplets is crucial to ensuring fast and good training. Generating easy triplets will result in slower convergence. Thus, we want triplets that do not meet the condition in Equation 1.5. There are three types of triplets:

Easy triplets: the triplets loss function is 0, because $d(a,p) + \alpha \ < d(a,n)$

Semi-hard triplets: the negative sample is not closer to the anchor than the positive, but it is within the margin. $d(a,p) < d(a,n) < d(a,p) + \alpha$

Hard triplets: the triplets negative sample is closer to the anchor than the positive. $d(a,n) < d(a,p)$

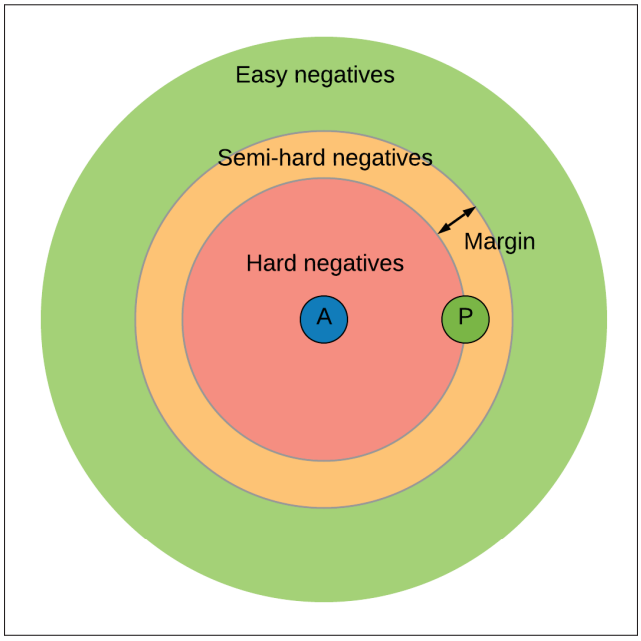


Figure 1.5 Triplet difficulty distribution.

To generate these triplets, we can use offline or online mining. Offline mining consists of generating all the embedding before each instance (epoch or step, for example). Then the hard and semi-hard triplets are kept for training. As the paper FaceNet (Schroff *et al.*, 2015b) describes, this methodology is not efficient, so they introduce online mining. The main idea is to generate the triplets at each mini-batch. There are two strategies for online mining which are

batch all and batch hard. In the batch all strategy, every valid triplet is selected and the loss is the average of the hard and semi-hard triplets. In the batch hard strategy, only the farthest positive and the nearest negative are selected from each sample/anchor. The generated triplets will be the hardest in the batch.

Center Loss. The center loss objective is to learn a center for each class (Wen, Zhang, Li & Qiao, 2016). It brings the sample representations near to each other by penalizing the distance between them and their corresponding centers. This loss function is good at reducing intra-variance. The center loss is defined as follows:

$$L_{center} = \frac{1}{2} \sum_{i=1}^{M} \left\| x_i - c_{j_i} \right\|_2^2 \tag{1.6}$$

The term x_i represents the representation of M total representations of the j_i class with i representing the depth of the representation. c_{j_i} is the center of the class j_i.

1.3 Face Recognition

The face recognition domain can be divided into multiple tasks. In this section, we review face detection, face tracking and face matching. Face detection techniques are required to capture the region of interest and to feed this region into the identification process.

1.3.1 Detection

The first efficient face detection technique was the Viola-Jones (Viola & Jones, 2001). This technique has the advantage of being computationally fast, but lacks in robustness when used in an unconstrained environment. Some of the state-of-the-art deep learning detectors are the Faster RCNN (Ren, He, Girshick & Sun, 2015), the Single Shot Multibox Detector (SSD) (Liu, Anguelov, Erhan, Szegedy, Reed, Fu & Berg, 2016) and the You Only Look Once (YOLO) (Redmon, Divvala, Girshick & Farhadi, 2016).

Popular single-shot detectors are the YOLO and the SSD. These architectures are also based on any CNN feature extractor (FE) architecture but they add a bounding box layer generator to the end of the FE. These methodologies require less computational resources, but give lower detection results. The MTCNN (Zhang, Luo, Loy & Tang, 2016) architecture directly addresses the problem of face detection. This is a strategy based on a cascaded architecture of three specialized deep convolutional networks. This methodology obtain key points from the faces such as the eyes, nose, and mouth. In this way, it is possible to train a face recognition model based on specific parts of the faces or to align it.

Faster RCNN

Based on its predecessors RCNN and Fast RCNN, Faster RCNN operates with a region proposal network (RPN) which is based on a CNN backbone. The RPN can be based on any CNN architecture. To produce regions, they add a small network onto the last feature map of the CNN that takes an $n \times n$ input. This small window slides over the feature map and flattens the feature map into a vector. This vector is then passed to two fully connected networks. One network is for the box regression proposal and the other network is for classification. For each window, they set a number of k anchors of different fixed sizes which make an output of $2 * k$ for object or not object prediction and $4 * k$ for bounding box regression.

A classifier is then passed over all of the proposed regions to produce a prediction score for each box. This architecture gets competitive performances on general object detection datasets, but this process can be rather slow compared to a single-shot object detection algorithm.

1.3.2 Matching

Casual object recognition models have a fixed number of classification outputs when trained using softmax cross entropy loss. However, in the case of face recognition, we may need to learn a variable number of subjects. To resolve this problem, Google introduced FaceNet (Schroff *et al.*, 2015b), a face recognition model trained using triplet loss. This model learns metric

representations of faces, where each representation can be compared to the others to measure the similarity.

Currently, there is a lot of research on face representation based on still face images that is very successful. This research generally uses deep CNNs and Siamese Architecture. Unfortunately, these methodologies are not as reliable when used with video-based face images. Thus, other researchers have proposed more complex methodologies and architectures to enhance the performance, but this has come at the cost of increased computational complexity and operational time, which can cause problems for real world applications. The sate-of-the art methodologies require information that is not available during operation or can only be obtained at a high price. It is not practical to implement those solutions in real world applications or to calibrate existing methods for use on newly installed video source such as cameras.

In (Ding & Tao, 2018), the authors used a complex ensemble structure that is not optimized for real-time operation. Also, it required two expensive details: facial landmarks which may fail due to occlusion and a lot of annotated images from the target domain for calibration. (Parchami, Bashbaghi & Granger, 2017b) proposed Haar-like features, and (Parchami, Bashbaghi & Granger, 2017a) proposed a more efficient network structure. However, both again required a large amount of annotated images with synthetic generation of video face samples from the target domain. Again, in (Wen, Chen, Cai & He, 2018), labeled data from the target is necessary for calibration.

One way to perform calibration automatically without labeled data from the target domain is to employ self-supervised learning. To do this, some use the spatio-temporal information (tracklets) given by a tracking algorithm performed on a video feed (Sharma, Tapaswi, Sarfraz & Stiefelhagen, 2019) (Wu, Lyu, Hu & Ji, 2013) (Cinbis, Verbeek & Schmid, 2011). In (Sharma *et al.*, 2019), this provided reasonable performance gain for large video files (a TV series episode) but did not perform well when there was not much unlabeled data available. It is important to note that these methods make the following assumption to label their face samples as either the same or different: samples within the same tracklet are the same and samples from two co-occurring tracklets are different. In the vast majority of cases, this hypothesis should be true.

1.4 Unsupervised Domain Adaptation

There has been a lot of work to try to bring face representations of the same identity as close as possible in the feature space while trying to move representations from different identities as far apart from each other as possible. Recent work has triplet loss and Siamese networks with few-shot learning and with few data (Hoffer & Ailon, 2015) (Laradji & Babanezhad, 2018). Training a robust face feature vector requires a large amount of data, which may not be available. This is why some have attempted to use Unsupervised Domain Adaptation (UDA) methodologies to adapt already existing robust models to new applications.

The UDA methods are more appealing and accessible, since they allow to transfer the knowledge across the domains without the need of labels from the target domain. From a global perspective, there are two directions of UDA based on adversarial learning (Goodfellow, Pouget-Abadie, Mirza, Xu, Warde-Farley, Ozair, Courville & Bengio, 2014); feature-level adaptation and pixel-level adaptation. For feature-level adaptation, it attempts to map the source and target domain in a same feature space to then use a classifier trained on that space that can work for both domains. For pixel-level adaptation, it transforms the images of the target domain to reassemble the images from the source domain, where a CNN trained on the source domain can classify images from the target domain.

Already, UDA for metric learning is being explored (Laradji & Babanezhad, 2018), (Sohn, Shang, Yu & Chandraker, 2019). The research M-ADDA (Laradji & Babanezhad, 2018) use adversarial learning (Ganin & Lempitsky, 2014) with simultaneous magnet loss to reduce the gap between domains while keeping the class center aligned. However, this methodology can only be applied on close-set problems where the classes are the same between the source set and the target set. In (Sohn *et al.*, 2019), the authors operate on open-set problems by introducing separation loss, which separates the source and target representations.

To perform UDA between a source and target domain, the authors of the methodology M-ADDA (Laradji & Babanezhad, 2018) propose a metric-based domain adaptation in two steps. Firstly, with the source dataset, they train a model with triplet loss to regroup the representations of the

same identity together and to separate the representations from different identities. This produces a powerful source model that is used for the next step. Secondly, they simultaneously attempt to confuse the representations from the source and target dataset with adversarial learning and to regroup the identities representations from the target dataset with magnet loss. To confuse the source and target representations, they use a discriminator that tries to distinguish representations from the source or target dataset while training the target model to produce representations that look like the source. At some point, they train the target model with magnet loss that pull representations from the same identities to their centers, limiting this part of the training to a limited number of identities. It is possible to only use the adversarial approach or the magnet loss training to perform UDA, but the authors noted that the UDA works best with both methods. In their research, they only experiment domain adaptation between the dataset MNIST and USPS, two written digits datasets with numbers from zero to nine. Applying this methodology for more complex problem like FR could give uncertain results.

The authors of (Sohn *et al.*, 2019) directly attempt UDA on face recognition problems. More specifically, they try to close the domain gap between still and video captured faces. They mention that their methodology requires a lot of unlabeled video images and labeled still images to work. Like the method describe above, they firstly trained a powerful source model with a source dataset made of, in their case, of annotated still images. To perform UDA, they do data augmentation with image transformation and adversarial learning. They apply transformations to the source images to make them like as if they were from a video and attempt to make the source model to reproduce the same representation as if there were no transformations. They note that it is not always possible to know all the variation factors between the two domains and use adversarial learning to address this. Again, like the method describe above, they use a discriminator to distinguish the images coming from the source and target dataset to attempt to align the two domains in the representation space. But, their discriminator is also able to discount the contribution of extremely noisy images to help with the training. They experiment on YouTube Faces and IJB-A datasets and achieve state-of-the-art accuracy.

CHAPTER 2

EXPERIMENTAL METHODOLOGY

In this chapter, we will survey the face recognition datasets and the evaluation measures used in this thesis. Some datasets are used for training and others are used for evaluation, in order to compare our results to the literature. The evaluation metrics have been chosen based on those used in previous research.

2.1 Datasets

In this section, we will review and describe the datasets used in our experiments. We used VGGface2 to train our base FE and LFW to benchmark it. To experiment with domain adaptation, we used COX-S2V and the TV series The Big Bang Theory.

2.1.1 VGGface2

For the training of the base feature extractor, we used VGGface2 dataset (Q. Cao *et al.*, 2018) which is a face recognition dataset. It is a large dataset composed of 9131 subjects annotated in 3.31M still images. This dataset is not known to have many identities, but to have many samples per identity, which is good for metric learning that requires many samples of the same identity. Figure 2.1 shows some examples of the image samples.

To facilitate recognition, we applied some pre-processing steps before the experiment. First, we removed unnecessary image background by applying MTCNN to each image to isolate the face. Then, each face is resized to an image of 160 x 160 pixels and saved as a JPEG file.

2.1.2 LFW

The validation of the base FE is performed on the LFW dataset (Wen *et al.*, 2018), a large dataset composed of 1680 annotated subjects with more than 13 000 still images. Like the training set,

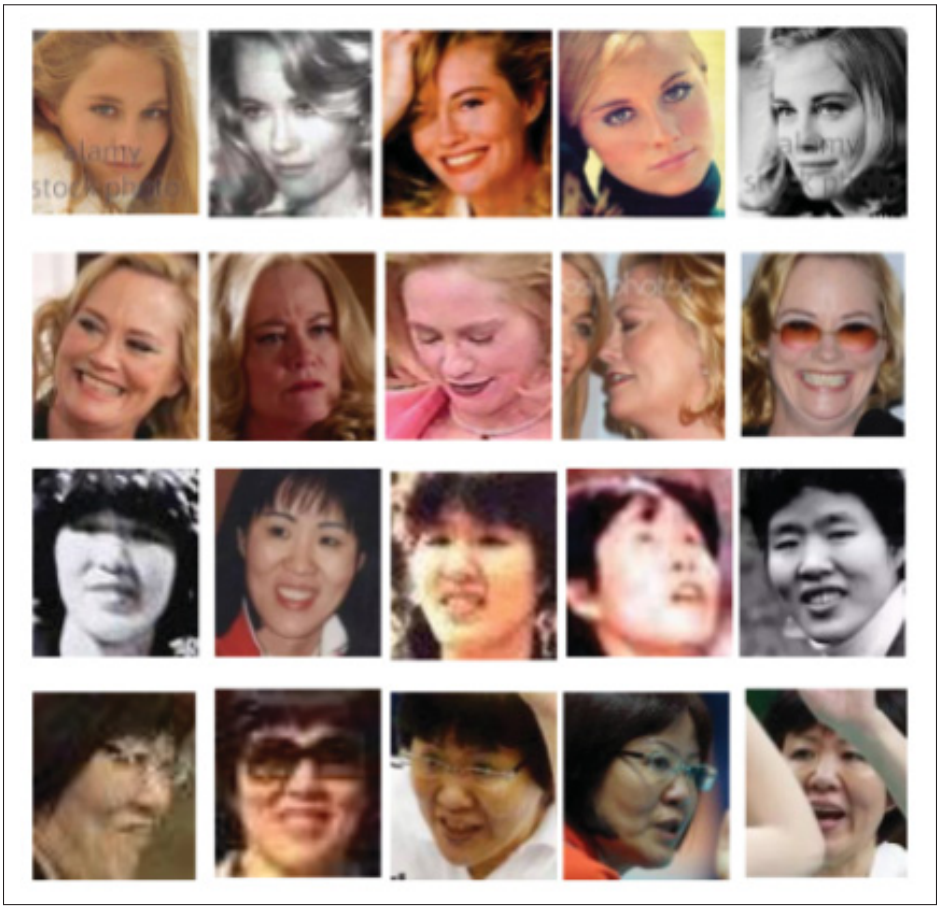


Figure 2.1 VGGface2 samples

we apply MTCNN to every image of the validation set to extract the faces and then resize them to 160x160 pixels. This dataset is widely used for benchmarking.

2.1.3 COX-S2V

Our methodology is evaluated using the COX S2V dataset (Huang, Shan, Wang, Lao, Kuerban & Chen, 2015). For each of the 1,000 subjects, there is one high quality still image and four video sequences captured by three cameras, making 4000 video samples in total. The videos capture conditions are as follows: the subject must walk an S-shaped parkour path (Figure 2.2) while being filmed by three different cameras, which produces four video sequences from four points of view as shown in Figure 2.4.

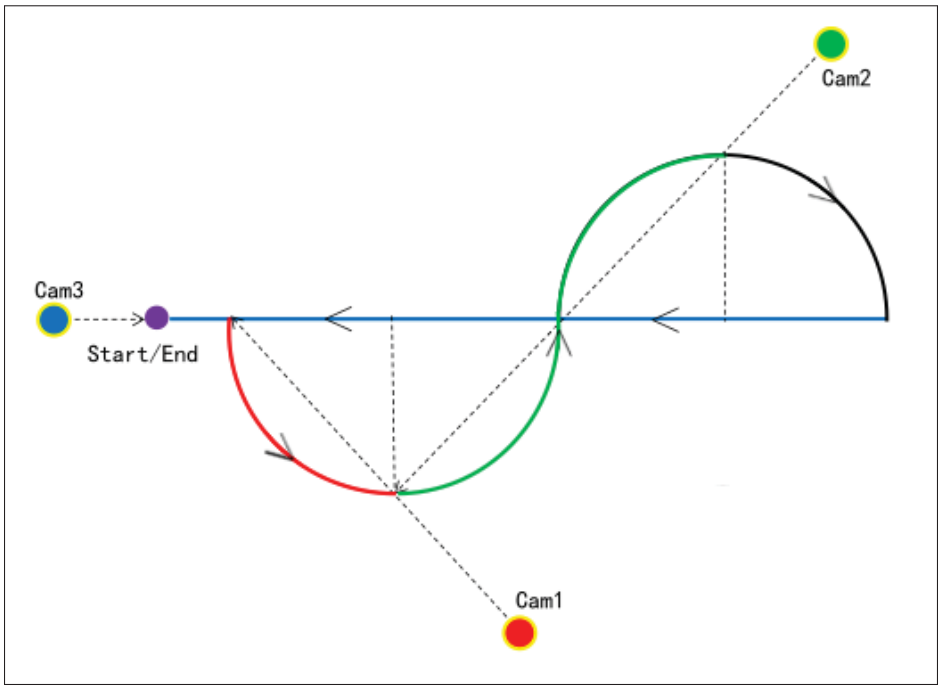


Figure 2.2 COX-S2V capture path (Huang *et al.*, 2015).

Figure 2.3 COX-S2V still sample.

a) CAM1/video4 b) CAM2/video3

c) CAM3/video2 d) CAM3/video1

Figure 2.4 COX-S2V video samples.

Following the evaluation protocol of the COX dataset paper (Huang *et al.*, 2015), the dataset is split with 700 identities used for evaluation and 300 identities used for training and validation. This allows direct comparison of our results to those of existing techniques like VGG-Face, TBE-CNN, HaarNet, and CCM-CNN. The COX dataset is usually used to evaluate the ability of a face recognition application, known to work on existing cameras, to work on a newly installed camera.

2.1.4 The Big Bang Theory TV series

To attempt video description for faces using our methodology, we used the first episode of the TV series The Big Bang Theory Season 1 which has been tested by other research. For comparison, we use the annotations from CVPR 2012 (Tapaswi, Bäuml & Stiefelhagen, 2012). These consist of bounding boxes that fit the characters' faces at each frame of the video. Each bounding box is associated with a label. For our experiment, we extract the region-of-interest defined by the bounding boxes and take the labels as ground truth for evaluation.

There are seven annotated characters that appear during the episode. The characters that appear most often are Sheldon, Leonard and Penny. The characters Howard, Raj and Kurt do not appear as often and have less samples. In the case of Kurt, he only appears in the title presentation. The CVPR 2012 annotations lastly annotated other characters as unknown which include the secretary at the beginning of the episode and children near the end. The imbalance in character samples makes the challenge harder to group identities.

2.2 Performance Measures

In this section, we will review two types of evaluation metrics, which are the classification metrics and clustering metrics. The classification metrics are used in Chapter 3 to evaluate the model prediction for sample pairs as being the same or not the same. The clustering metrics are used in Chapter 4 to evaluate the quality of the predicted identity groups.

2.2.1 Classification Measures

The performance evaluation metric of a Siamese network can be similarly evaluated as a binary classifier where the prediction is the same or not the same. When comparing representations in order to measure the similarity, they are classified as the same if the similarity is over a certain threshold. A high similarity threshold will give high confidence for same label, but will augment the risk of labeling positive pairs as not the same if the samples have different capture conditions that make their representations more different from each other. A low similarity threshold should label more of these hard positive pairs as the same, but at the cost of labeling more negative pairs as the same.

It is important for these metrics to balance negative pairs and positive pairs. The evaluation pairs are generated once and saved to a file which will be used for the evaluation of all the experiments, giving a fair comparison between the experimentations.

Accuracy

The accuracy is the proportion of correctly labeled predictions on the total number of predictions. On a binary and balanced evaluation set, an accuracy of 50% means that the predictions are random, while 100% means that the predictions are perfect.

The terms of the calculation of accuracy are as follows:

- **TP**: True Positive are positive pairs labeled as the same.
- **TN**: True Negative are negative pairs labeled as not the same.
- **FP**: False Positive are negative pairs labeled as the same
- **FN**: False Negative are positive pairs labeled as not the same.

The accuracy is calculated as:

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.1}$$

The equation terms are calculated with a distance threshold optimized on the validation set to maximize accuracy. In the case of LFW, the evaluation set is divided into 10 equal subsets and the distance threshold is the mean of the optimization results on each of these subsets.

Validation Rate

In this thesis, the validation rate (VAL) is defined by the FaceNet research to which we will compare our results. The validation rate is the True Acceptance rate (TA) but at a distance threshold defined by a fixed False Acceptance Rate (FAR). All same identities pairs are defined by ρ_{same} and all pairs of different identities are defined by ρ_{diff} The terms are defined below.

$$VAL(d) = \frac{|TA(d)|}{|\rho_{same}|},\ FAR(d) = \frac{|FA(d)|}{|\rho_{diff}|} \tag{2.2}$$

We compute the validation rate at FAR=0.001. The distance threshold is calculated with the FAR equation and then the validation rate is calculated with that threshold. This metric allows us to better compare our results with those of the LFW dataset as the accuracy score of the state-of-the-art techniques are all near 99.9%.

2.2.2 Clustering Measures

Here, we will review some clustering metrics used in this work. Some of these metrics have been chosen for their ability to represent accurately the clustering quality in order to compare our methodologies with other research. We will review silhouette score, Weighted Purity and V-measure.

Silhouette Score

The silhouette score (Rousseeuw, 1987) does not require the ground truth to be calculated. It measures the similarity between a data point and its own cluster in comparison to other clusters. To compute the silhouette score, we first calculate the mean distance a(i) between a data point i and all the other data points j within the same clusters. That value represents the goodness of its assignment to its cluster.

$$a(i) = \frac{1}{|C_i| - 1} \sum_{j \in C_i, i \neq j} d(i, j) \tag{2.3}$$

Then, we find the minimum mean distance to all points in the other clusters. It can be viewed as the mean dissimilarity to the closest other cluster.

$$b(i) = \min_{k \neq i} \frac{1}{|C_k|} \sum_{j \in C_k} d(i, j) \tag{2.4}$$

Finally, the silhouette score is defined as follows:

$$s(i) = \begin{cases} 1 - a(i)/b(i)) & \text{if } a(i) < b(i) \\ 0 & \text{if } a(i) = b(i) \\ b(i)/a(i) - 1 & \text{if } a(i) > b(i) \end{cases} \tag{2.5}$$

A score near 1 signifies that the data corresponds to its own cluster while being far from other clusters, while a score near -1 signifies that the clusters are mixed and hardly separable.

Weighted Purity

The weighted purity (Tapaswi, Parkhi, Rahtu, Sommerlade, Stiefelhagen & Zisserman, 2014) is calculated by summing all of the clusters and dividing them by the total number of samples N.

The weighted purity is express in equation 2.2.2 where c is a cluster in the set of C clusters and k is a class in the sets of K classes.

$$purity = \frac{1}{N} \sum_{k \epsilon K} \max_{c \epsilon C} |k \cap c| \tag{2.6}$$

This metric has a few drawbacks. It does not penalize the score when there are many clusters or when small clusters are misclassified (imbalanced data). It is a good metric for evaluating the homogeneity of the predictions, but it does not evaluate the compactness.

V-Measure

The V-Measure (Rosenberg & Hirschberg, 2007) is a conditional entropy-based external cluster evaluation measure. It considers the homogeneity and completeness of the clustering. To achieve homogeneity, each predicted class must be present in only one cluster. The homogeneity is expressed as follows:

$$h = \begin{cases} 1 & \text{if } H(C|K) = 0 \\ 1 - \frac{H(C|K)}{H(C)} & \text{else} \end{cases} \tag{2.7}$$

Where

$$H(C|K) = -\sum_{k=1}^{|K|} \sum_{c=1}^{|C|} \frac{a_{ck}}{N} log \frac{a_{ck}}{\sum_{c=1}^{|C|} a_{ck}} \tag{2.8}$$

$$H(C) = -\sum_{c=1}^{|C|} \frac{\sum_{k=1}^{|K|} a_{ck}}{n} log \frac{\sum_{k=1}^{|K|} a_{ck}}{n} \tag{2.9}$$

To achieve completeness, all representation of a single identity must be classified into a single cluster. It is symmetric to the homogeneity. The completeness is expressed as follows:

$$c = \begin{cases} 1 & \text{if } H(K|C) = 0 \\ 1 - \frac{H(K|C)}{H(K)} & \text{else} \end{cases} \tag{2.10}$$

Where

$$H(K|C) = -\sum_{c=1}^{|C|}\sum_{k=1}^{|K|}\frac{a_{ck}}{N} log \frac{a_{ck}}{\sum_{k=1}^{|K|} a_{ck}} \tag{2.11}$$

$$H(K) = -\sum_{k=1}^{|K|}\frac{\sum_{c=1}^{|C|} a_{ck}}{n} log \frac{\sum_{c=1}^{|C|} a_{ck}}{n} \tag{2.12}$$

Finally, the V-measure is expressed as follows where β permits more weight to be given to the homogeneity score or to the compactness score.

$$V_\beta = \frac{(1+\beta) * h * c}{(\beta * h) + c} \tag{2.13}$$

2.2.3 Complexity

To compare the complexity of our method with others, we simply note the number of parameters and operations use in the methodology. We use these measures because they are independent of the hardware used. In this thesis, we base our experimentation on one CNN architecture the Resnet50. This CNN is used as a face feature extractor to produce face representations. The Resnet50 typically has around 23 millions parameters and has an inference time of less than 3 ms on an Nvidia Titan XP if used with a batch size greater than 1 (Bianco, Cadene, Celona & Napoletano, 2018). This architecture is heavier than other more lightweight, architectures such as mobilenet,

which have an inference time of less than 1 ms. However, it is not as complex as, for example, the Resnet-152, which takes two to three times more time to achieve the same performance with only slightly greater accuracy.

CHAPTER 3

FACE RECOGNITION WITH SIAMESE NETWORKS FOR VIDEO SURVEILLANCE

3.1 Introduction

Generating and calibrating discriminant representations of face images is crucial for new capturing devices and new capture environments. This is especially true for face recognition with video surveillance. The representation learning methodologies for still face images is well-developed as there are many large datasets of still images available. It allows human-level performance, like the current face recognition (FR) systems (Schroff, Kalenichenko & Philbin, 2015a) .

Face representation captured from video is harder to discriminate for two main reasons. First, the face is captured in an unconstrained manner that can lead to major variability in facial appearance, pose, illumination, scale, expression, etc. Second, there are significantly fewer video-based datasets available, compared to still-image datasets. There are not enough samples to learn reliable deep face representations. For example, the YouTube face dataset (Wolf, Hassner & Maoz, 2011) contains 3.4K videos with 1.5K different subjects while the still-image VGG face2 dataset contains 3.3M faces of 9K subjects (Q. Cao *et al.*, 2018).

One way to tackle this challenge is to reduce the variability of video-based images to a level similar to that of still images. Doing so will enhance the performance of the powerful still-image face representations on video face samples. For example, some use the autoencoder deep neural network to learn discriminant face representations with wich to produce high-quality images (frontal, well-illuminated, less blurred faces with neutral expressions) from video-based images (Parchami, Bashbaghi, Granger & Sayed, 2017c). This method requires a lot of data from the target domain and is not practical for the calibration of newly installed video sources.

Another way to approach the problem of learning face representation for video is to create video samples from still samples by adding effects similar to the video capturing conditions. Then, these samples can be used to train new face representations for the specific video application. For example, applying artificial blur to the still training data should compensate for the real world video application. A training dataset composed of still and blurred face samples can encourage a CNN to produce face representations insensitive to blur (Ding & Tao, 2018). Also, some have proposed recreating the face samples from specific capture variations (pose, illumination, etc.) to make them similar to samples captured under the conditions of the operational domain (Mokhayeri, Granger & Bilodeaun, 2019) (Hong, Im, Ryu & Yang, 2017). These methodologies are very complex and are not efficient for the calibration of new video capturing devices, and they also may not be capable of covering the complete range of capturing conditions.

Currently, face representation learning for video is addressed using domain adaptation (DA) methodologies. These attempt to adapt representation trained with high quality face images (source data) to a different environment and capturing conditions such as video samples (target data). With labeled data from the target domain, some methodologies perform a supervised fine-tune of the source model (Wen *et al.*, 2018), but obtaining these labeled samples is not practical. More efficient methods of enhancing the video representations use Unsupervised Domain Adaptation (UDA), which adapts the source model using unlabeled data from the target domain (Wen *et al.*, 2018) (Luo, Hu, Deng & Shen, 2018) (Ganin & Lempitsky, 2014). However, these methods are not designed for open-set problem where the number of classes is unknown. A Siamese network is able to address this problem by measuring the similarity between face representations.

Recently, works on UDA for deep metric learning were put forward (Laradji & Babanezhad, 2018)(Sohn *et al.*, 2019), but these methodologies are designed for closed and small-set challenges like handwritten digit recognition (Laradji & Babanezhad, 2018), or use multiple techniques for more difficult problems making the implementation inefficient for other applications such as automatic calibration for video surveillance (Sohn, Liu, Zhong, Yu, Yang & Chandraker, 2017)(Hong *et al.*, 2017).

For this specific case of UDA where the adapted model is distance metric based rather than a feature based model, we believe that the loss function should be optimized in the distance space rather than the feature space. By doing so, we try to differentiate distances that are within class (positive pair) or between class (negative pair). Also, this metrics works with close-set and open-set problems, since it work with pairwise distances like some existing methods (Laradji & Babanezhad, 2018)(Sohn *et al.*, 2019). We use this concept to design our proposed method.

Firstly, we capture a raw frame from the scene. The frame is fed into the face detector, which is a deep learning model that scans the whole images for visible faces. This is a slow process as the state-of-the-art detection technique used is computationally expensive. It generates a bounding box for each detected face and sends it to the face tracker algorithm.

In general, tracking algorithms require low computational resources as they operate only in regions of interest of the images. The tracker creates trajectories of the same entity over time, feeding samples to the face recognition system.

The face recognition system is based on the Siamese net architecture. It extracts feature vectors from the reference samples and stores them in a database. Using the same feature extractor, it extracts features from the samples to identify and then compare to the reference features. The feature extractor is based on the FaceNet model which is trained using triplet loss. It produces representations where distances between two inputs correspond to the similarity between them. This methodology permits adding a person of interest to the database without retraining a classifier.

3.2 Face Recognition for Video Surveillance

In this section, we describe the different components of our face recognition system for video surveillance. For face segmentation, we use FAST-DT (FAce STructured Detection and Tracking), a methodology that combines detection and tracking techniques to produce face trajectories. Then, the samples for these trajectories are given to a Siamese network to identify the face.

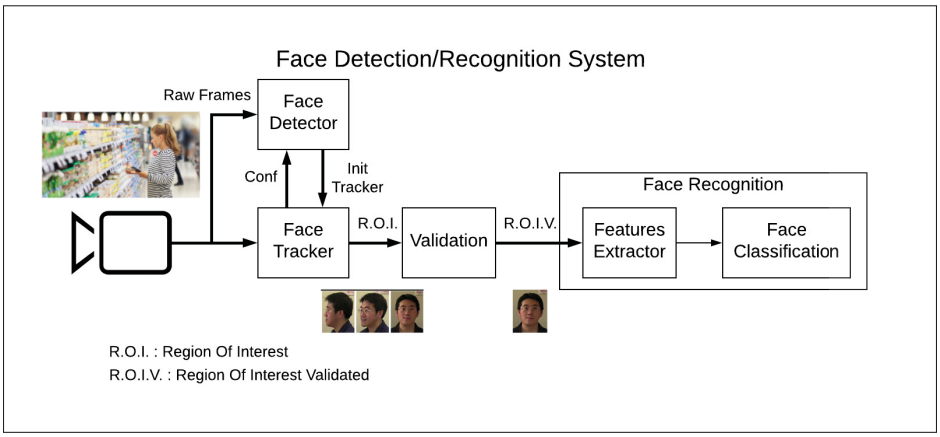


Figure 3.1 Face recognition system for video surveillance.

3.2.1 Detection and Tracking

FAST-DT (FAce STructured Detection and Tracking) (Comaschi, Stuijk, Basten & Corporaal, 2015) produces bounding boxes around faces in video. The idea behind this structure is to offer a subject trajectory by coupling a detector algorithm with a tracker algorithm. This permits the gathering of more samples of the regions of interest in a given time lapse. In general, the detector will have to input the whole frame, making it slower compared to the tracker, which only operates around the region of interest. By running the detector only at each N frames and the tracker at each frame, we achieve greater time performance and more samples.

The operation is as follows:

1. The face detector operates at each N frame on the raw frame. It feeds bounding boxes to the SORT function.

2. The SORT instance compares the actual tracking with the new detection bounding box (bbx).

    a. If the detection bbx does not corresponds to one of the actual tracks, SORT initiates a new tracker.

    b. If the detection bbx correspond with one of the tracks, SORT updates that track.

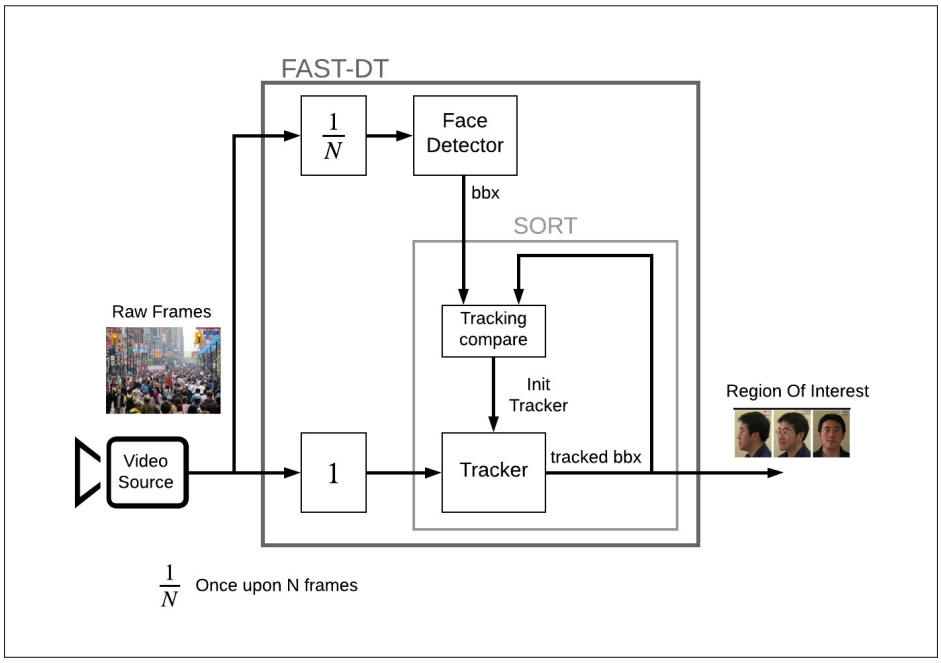


Figure 3.2 FAce STructured Detection and Tracking

c. If there is no detection bbx that corresponds with an actual track, SORT terminates that track. However, there is an option to keep the tracking for a number frames, which would allow to connect face trajectories if the face detector failed to capture the same face during some frames.

3. The tracker operates at each frame giving the resulting samples from each track.

3.2.2 Pair-wise Matching

The face recognition module is able to identify a person using only a single reference in its database. The module requires a preparation phase prior to operation to save computational time. In the preparation, we extract the features from the target person's database. These features are stored in an embedded database, ready for comparison. Using the same feature extractor used in the preparation phase, the module extracts the features from the persons of interest. The features are then compared to each of the reference features using a similarity matcher. If the similarity between the two feature vectors is high, identification is achieved.

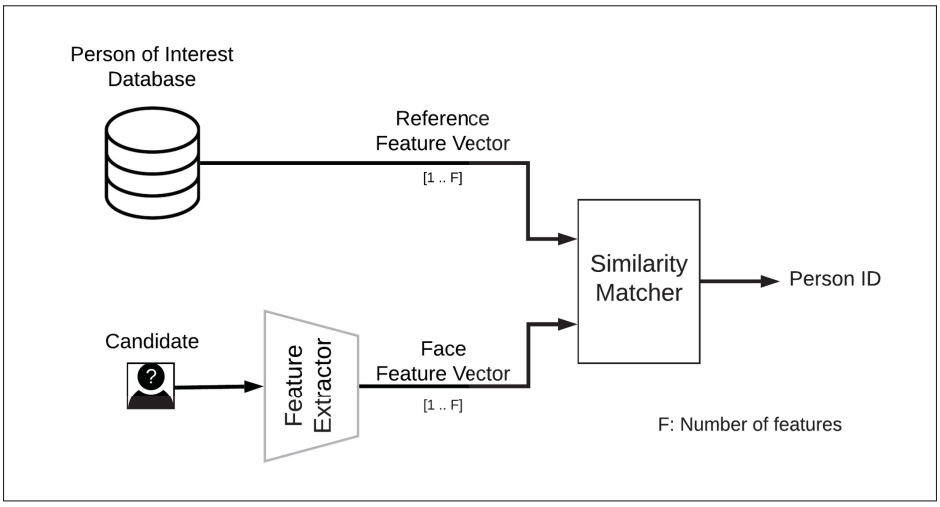
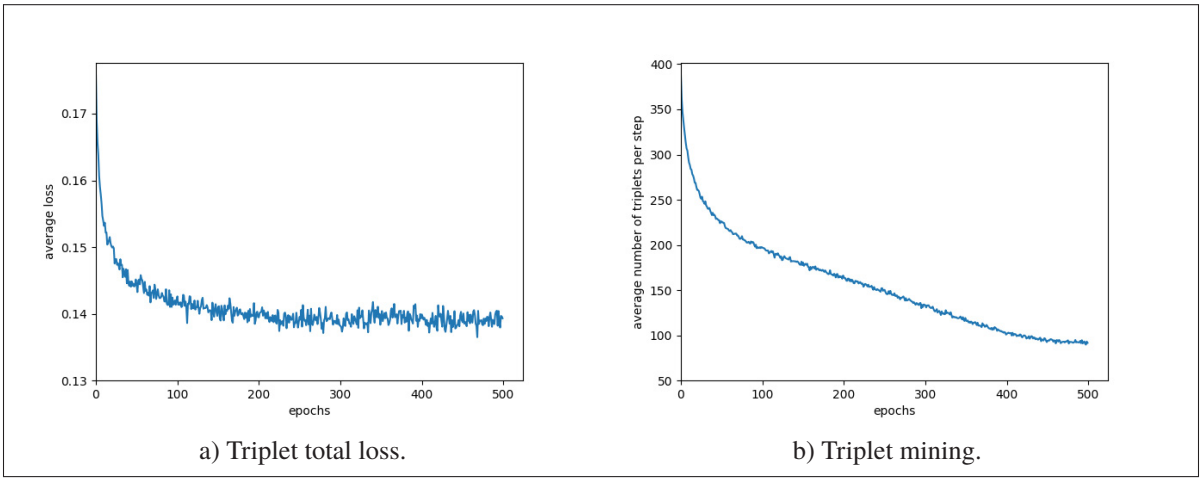


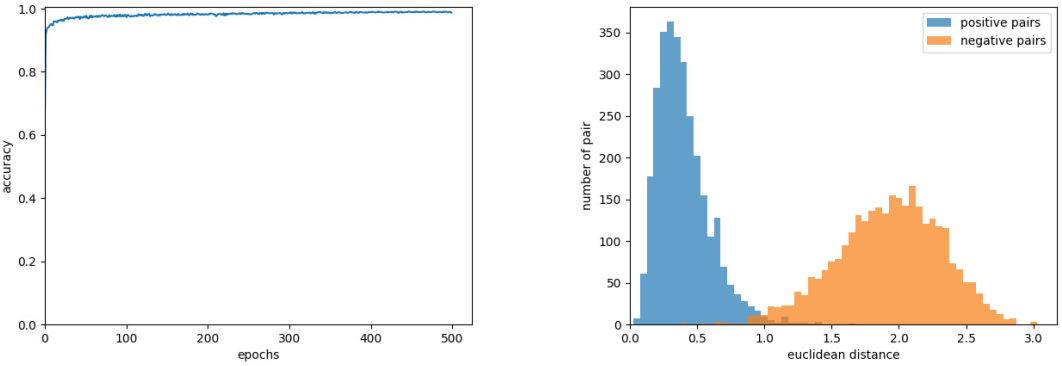
Figure 3.3 Face identification with a Siamese network.

The hyperparameters and the results of the training of the base FE are listed here. Using triplet loss, we trained a ResNet50 architecture, which was pre-trained on the ImageNet dataset with the VGGface2 dataset. For each epoch, 40 subjects and 5 samples per subject are randomly fetched on which to apply the semi-hard mining strategy. Each sample is randomly flipped horizontally. We set up a cosine annealing learning rate decay which starts at 0.01 and finishes at 1e-6. The model was validated on the LFW dataset at each epoch.



a) Triplet total loss. b) Triplet mining.

Figure 3.4 Base FE training metrics.



a) LFW validation accuracy during training. b) LFW distance pairs distribution.

Figure 3.5 Base FE evaluation metrics on LFW.

Table 3.1 Base FE LFW evaluation metics.

metrics	**ours**	**FaceNet**
accuracy	99.06% ± 0.53%	98.87% ± 0.15%
validation rate FAR=0.001	87.70% ± 3.32%	87.90% ± 1.90%
AUC	0.99915	-

3.3 Supervised Domain Adaptation of Video Face Representations

From the pre-trained feature extractor on VGGface2, we fine-tuned the model with COX S2V dataset to improve its ability to generate discriminative representations from video samples. To simulate our problem, we select viewpoints from COX videos as the source domain and set another viewpoints as the target domain. The source videos represent the training set where samples are annotated and used for supervised learning using classic triplet loss. The target videos represent the viewpoints where we want to fine-tune the feature extractor without annotating the samples. The samples are used for producing an adversarial loss that will shorten the distance between target and source domains.

First, we fine-tune the FE on COX-S2V video samples from the chosen source viewpoints using classic triplet loss. Then, we pursue the learning by adding the target triplet term produced by

the target samples. As we do not know the target classes, the target samples are chosen randomly within the training distribution. The following list summarizes the methodology steps:

- The FE is trained on VGGface2 using triplet loss.
- The FE is trained on 300 subjects of COX S2V using triplet loss.
- The FE is evaluated on the other 700 subjects of COX S2V.

Table 3.2 Base feature extractor on COX S2V datasets.

Model	video 1	video 2	video 3	video 4
Base FE	75.80±2.41	94.05±1.14	68.57±2.06	91.97±1.50
Supervised fine-tune	96.70±1.72	98.12±0.93	95.13±1.54	97.82±0.91

3.4 Unsupervised Domain Adaptation of Video Face Representations using Dual-Triplet Metric Learning

This section addresses the challenges of adapting face representations from video to a new video source or environment, using only unlabeled target data so that the methodology is practical and efficient. This chapter is from the same work of the paper Unsupervised Domain Adaptation of Video Face Representations using Dual-Triplet Metric Learning from IJCNN 2020 (Ekladious *et al.*, 2020). The contributions of this work are as follows:

- we introduce a new domain adaptation framework called Dual-Triplet Metric Learning (DTML) that allows adaptation of a metric to a target dataset using unlabeled data from the target domain.
- we propose a mutually-supervised learning method where the source (teacher) data labels the unlabeled target (student) data.
- we apply the proposed DTML and mutually-supervised methods to the still-to-video face recognition application and provide a level accuracy comparable to state-of-the-art methods for video face representation learning, but with unsupervised domain adaptation capability.

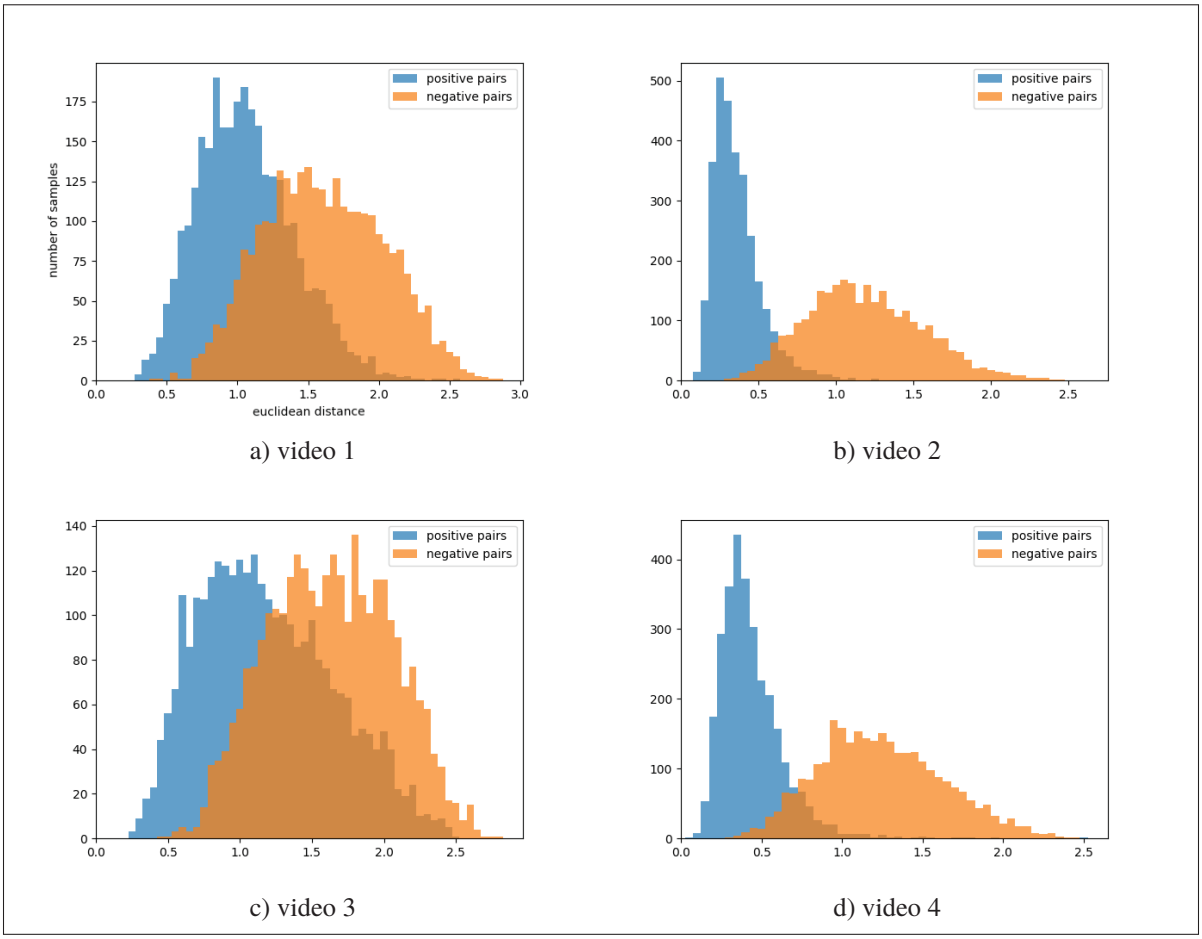


Figure 3.6 COX-S2V distance distribution with base feature extractor.

It is important to mention that the proposed DTML with mutually-supervised learning can be applied to different modalities, while in this paper we assess the method using the video-based face recognition as a specific use case.

3.4.1 Dual-Triplet Metric Learning

We propose Dual-Triplet Metric Learning (DTML) as a framework for domain adaptation, as illustrated in Figure 3.7. This framework offers an easy and efficient way to calibrate new video sources with an existing face recognition video system such as a video surveillance network. The methodology can be used to perform domain adaptation on existing model to a new capturing device in an environment different than that of current operational devices.

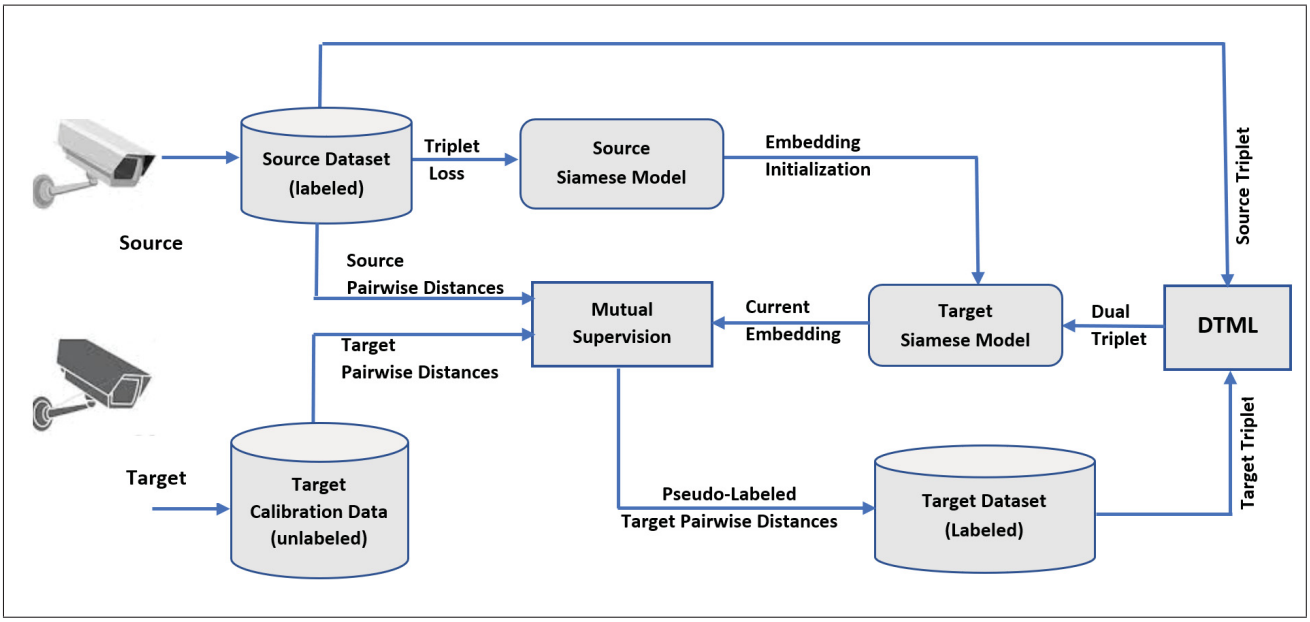


Figure 3.7 Dual-triplet metric learning for domain adaptation

The calibration data is composed of faces with unknown identity from a newly installed capturing device that we will refer to as target data. The faces must be extracted from the video images using face segmentation techniques. The capturing conditions from this new device will be represented in these captured Regions of Interest (ROIs). The source data will be considered the teacher during the learning process while the target data will be considered the student. The teacher will provide the student with its initial knowledge acquired through supervised learning. During learning, the student and teacher will share their knowledge using the labeled data from the source and the unlabeled data from the target.

An initial face representation is learned using classic triplet loss methodology (Schroff *et al.*, 2015a). Then, given that a new capturing device is installed, the source representations is enhanced for this new device with the target samples by minimizing the dual-triplet loss. In this process, one triplet set from the source with one triplet set of target is used. To form the target triplets, we compute the pair-wise distance of a batch of target samples using the source model and label each target distances as being within-class or between-class. The idea is that during learning, the source and target representations distributions become more similar. The

hypothesis behind this is that the distance metric used for the source domain should work in the target domain. Also, the classification of a pair of target samples as being within or between should be as accurate as a pair of source samples.

3.4.2 Dual-Triplet Loss

The dual-triplet loss L consists of two terms: a source term L_s and a target term L_t:

$$L = L_s + \lambda . L_t. \tag{3.1}$$

where λ is the parameter that balances the two objectives.

A source triplet L_s is constituted from labeled source data, using an anchor a, a positive sample p, a negative sample n, and a margin α:

$$L_s = max(||f(a) - f(p)|| - ||f(a) - f(n)|| + \alpha, 0) \tag{3.2}$$

To create target triplets from unlabeled data, we compute the distance matrix of a batch of these unlabeled samples and classify the pairwise distance as being either within-class (wc) or between-class (bc). From a chosen anchor, the within-class representation are candidates for positive pair and the between-class are candidates for negative pair. The target triplet is constituted as follows where $||wc||$ and $||bc||$ is the within-class and between-class distance respectively:

$$L_t = max(||wc|| - ||bc|| + \alpha, 0) \tag{3.3}$$

The objective of the source triplets is to offer a robust distance metric since it was learned with labeled data from the source domain. For the target triplets, the objective is to attempt to

separate successfully the positive pairs from the negative pairs within the defined margin. In the end, the pairwise distance distribution of positive and negative representations should become similar between source and target.

Note that the proposed DTML can also work with labeled target data if available. For example, co-occurring face trajectories can hypothetically form negative samples while face samples of the same trajectories can be positive samples. With this information, it would be possible to form triplets but it requires co-occurring face tracks, which are not guaranteed in every surveillance application. This is why we introduce the mutually-supervised method that will attempt to form triplets from unlabeled data.

3.4.3 Mutual Supervision

The proposed mutually-supervised learning method is intended to label the target samples with knowledge from the source model. In other words, the method will attempt to form positive pair samples (within-class) and negative pair samples (between-class) with the unlabeled target samples.

Figure 3.8 illustrates the mutual supervision learning using the distributions of distances between samples from same-person (within-class (WC) samples) and distances between samples from different-persons (between-class (BC) samples). Left column (a,c) show the distance distributions for the source data (teacher) and right column (b,d) show the distributions for the target data (student). Upper row (a,b) show the distance distributions for the initial source representation, while the bottom row (c,d) show the distributions where a shared target representation is learned using the proposed dual-triplet and mutual-supervision learning.

The process of learning target representations is as follows: A batch of training samples from the source and target data are passed through the pre-trained feature extractor. Then, we compute the Euclidean distance matrix for the target and source representations separately. With the pairwise distance found with the source labeled data, it is possible to find the distributions of pairwise distances as being within-class (WC) or between-class (BC) (see Figure 3.8.a). These

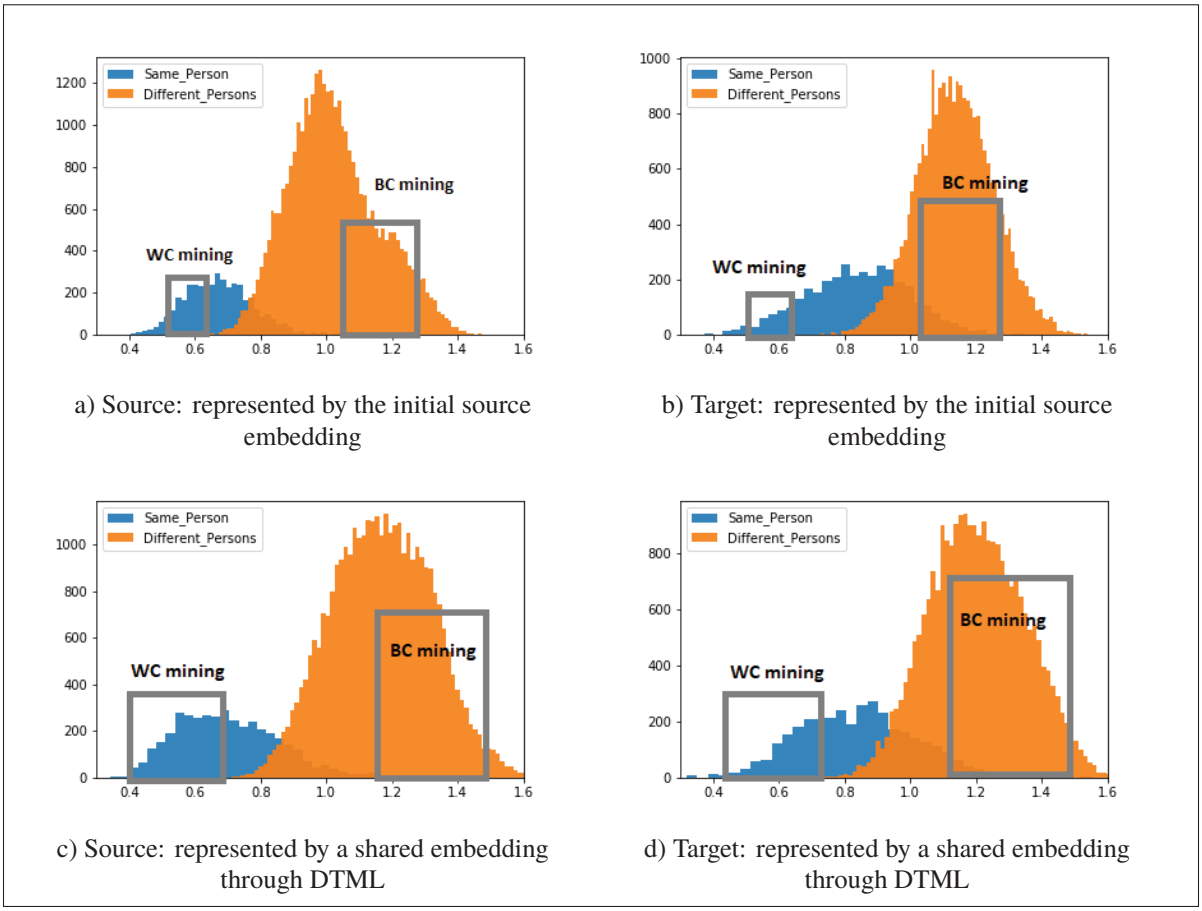


Figure 3.8 Illustration of the mutual supervision learning

distributions are finally used to label the target samples using two mining windows: 1) the within-class mining window (WC_{mw}) and 2) the between-class mining window (BC_{mw}):

$$WC_{mw} = [\mu_{wc} - \sigma_{wc}, \mu_{wc}]. \tag{3.4}$$

$$BC_{mw} = [\mu_{bc}, \mu_{bc} + \sigma_{bc}]. \tag{3.5}$$

where μ_{wc}, σ_{wc} and μ_{bc}, σ_{wc} are the mean and standard deviation of the WC and BC pairwise distances, respectively.

These mining windows are computed with the objective of finding a compromise between mining easy pairs far from the confusion area that will not give much learning enhancement, and mining pairs within the confusion area that are at high risk of being incorrectly labeled but that will give the highest learning enhancement. It is important to note that pairs beyond the distance margin of the loss function will not contribute to the learning process.

With these defined window computed from the labeled source samples, the target pair distances are defined as within or between class (see Figure 3.8.b). In the beginning of this process, the source and target pairwise distance distributions may not be aligned as a consequence of the domain shift. There is a possibility that the domain shift is so severe that the mined samples are not accurate enough. These chosen samples from the target domain are randomly chosen within these windows to form the target triplets. With this, the dual triplet is formed by combining it with triplets from the source.

The described process is applied for each training epoch from which a batch of a samples is captured. After some time, the within and between class distributions should become aligned between the source and target pairwise distances and are separated by the same margin. Once this is achieved, it will be possible to use the feature extractor to produce discriminant features for both domains.

Figure 3.8 (c and d) show the alignment between the source and target distributions after DTML with our mutually-supervised training. The better the alignment, the more likely it is that the source distribution will correctly label the target pairwise distances as being within or between classes. It is in the vision of this methodology to compensate for the inaccuracies of the target triplets caused by the imperfection of our pseudo-labeling technique with the perfect source triplets since they are labeled.

3.4.4 Experimental Methodology

Our proposed DTML framework may be applied to many problems. In this contribution, we apply the methodology to a still-to-video (S2V) face recognition application. In this application,

	Cam3 → Cam1		Cam1 → Cam3		Cam3 → Cam2	
	AUC	Acc %	AUC	Acc %	AUC	Acc %
VGG-Face (Q. Cao *et al.*, 2018)	-	69.6	-	68.1	-	76.0
Source model: without DA	0.98	95.0	0.90	81.7	0.94	87.2
Proposed UDA: DTML-A	0.99	97.7	0.95	88.7	0.97	91.0
Upper-bound:(Supervised DA)	0.99	98.0	0.98	93.3	0.99	95.3
TBE-CNN (Ding & Tao, 2018)	-	87.8	-	88.2	-	95.7
CCM-CNN (Parchami *et al.*, 2017a)	-	87.8	-	88.6	-	92.1
HaarNet (Parchami *et al.*, 2017b)	-	87.9	-	89.3	-	97.0

Table 3.3 Results on the COX Face Recognition dataset using the proposed DTML method with mutual-Supervision.

video surveillance cameras capture scenes that contain people. Then, the face ROIs are captured from the images using face detection/tracking. These ROIs are compared to high quality frontal still face images from a database in order to find the identity of the captured person (if in the database).

For the COX dataset, this experiment follows the experimental protocol in (Huang *et al.*, 2015) to compare the results with the state-of-the-art techniques. The experimental protocol is as follows: from the 1000 subjects, 300 are used for training while the other 700 subjects are used for evaluation. To simulate a newly-installed camera, 200 of the training subjects are used to train the initial source model while the other 100 subjects are from the newly installed target camera. This split simulates a real operation case where the new camera will be calibrated using identities other than those used to train the model initially. Also, this split simulates an open-set case where the identities for the evaluation phase were not seen during training and calibration.

For all experiments, we start with our model trained with triplet loss on VGG-face2 dataset (Q. Cao *et al.*, 2018). This model is the starting point for fine-tuning and calibration, but also serves as the lower-bound of our experiments. This lower-bound represents the case if no calibration at all is performed on the model.

Our proposed UDA methodology is tested by using the training set from the source camera and the calibration set from the target camera to constitute the dual-triplet. Then, the DTML methodology is applied to enhance the model performance on the specific target camera.

To test the impact of the dual-triplet loss terms, three scenarios are implemented:

1. L_s: where only data from the source camera are used and only the source triplet is used to tune the network. This scenario simulates the case where we do not use calibration data and only improve the source representation.

2. L_t: where only data from the target camera are used and only the target triplet is used to tune the network. This scenario simulates the case where we only rely on calibration data to adapt the source model for the new camera.

3. $L_s + L_t$: where data from both the source and calibrated cameras are used for adaptation, which is the DTML proposed method.

The experiments are performed with a batch size of 100 source samples plus 100 target samples. In each source batch, we select 5 subjects and 20 samples per subject, which gives $20 \times 20 \times 5$ possible positive pairs and $20 \times 80 \times 5$ possible negative pairs. For the target batch, the mutual-supervision method is employed to acquire an equivalent number of positive and negative pairs. Once the sample batches are formed, the DTML runs for 40 epochs. The performance is evaluated with rank 1 accuracy and the Area Under ROC Curves (AUC). The parameter λ (see 3.1) is used to weigh the target triplet term in the dual-triplet loss function, but extensive experimentation has shown that $\lambda = 1$ give the best results.

3.4.5 Experimental Results and Discussion

The face recognition results for the dataset COS-S2V are shown in Table 3.3. They clearly shows that only performing supervised fine tuning with the base feature extractor on samples images from the same network enhance the performance. Already with this step, the domain shift gap

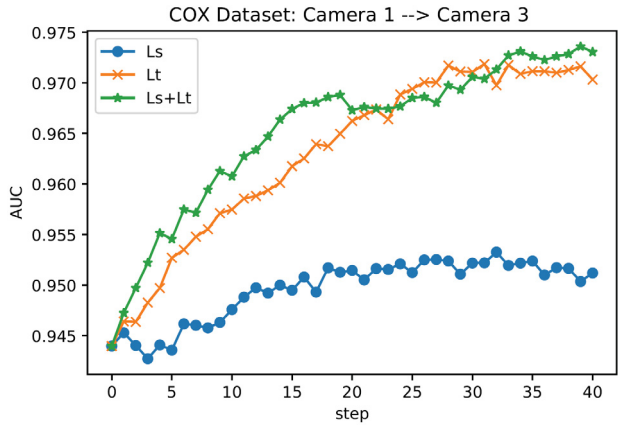


Figure 3.9 AUC performance on COX dataset camera 1 to camera 3

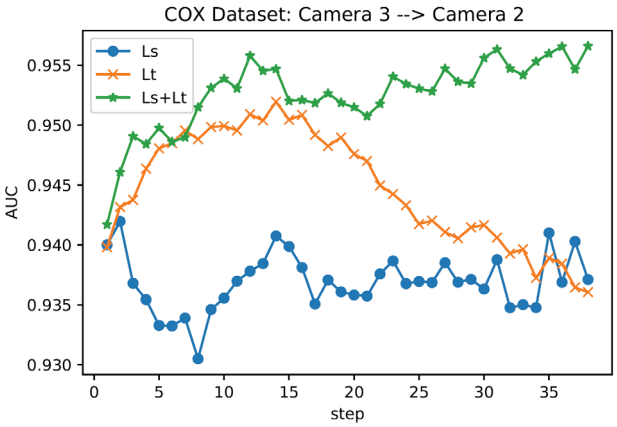
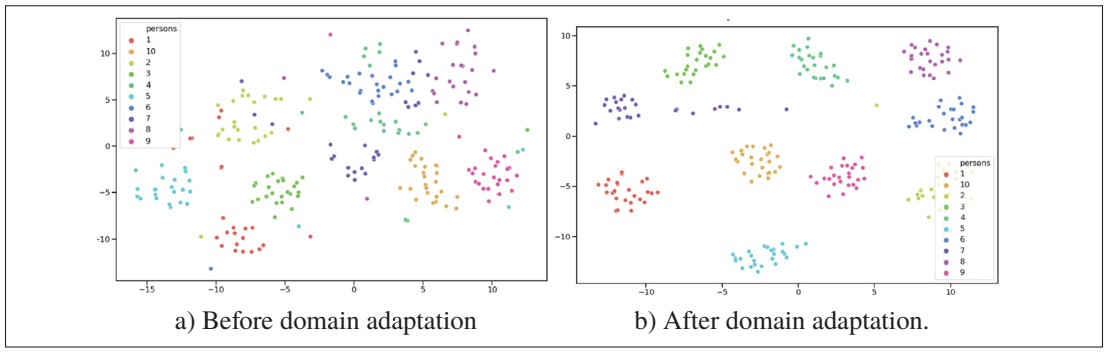


Figure 3.10 AUC performance on COX dataset camera 3 to camera 2

is reduced. This is expected, as the camera set-up for COX-S2V share many environmental similarities between each others.

Our proposed DTML algorithm with mutual-supervision gives a significant improvement of an approximately 5% increase for AUC to help reduce the domain shift. The DTML methodology could perform reliable unsupervised domain adaptation (UDA) and give performances near the upper bound of this experiment.

The proposed method gives similar result in some cases (Cam3 $\rightarrow$ Cam2) to state-of-the-art methods but performs less well in other cases (Cam1 $\rightarrow$ Cam3). It is worth noting that these



a) Before domain adaptation b) After domain adaptation.

Figure 3.11 COX dataset: TSNE representation for the first ten people in the testing set

techniques are less practical for two reasons. Some are less efficient because of the use of complex ensembles of CNNs or expensive generation of synthetic video face samples. Also, some of these methods require information to perform calibration, such as facial landmarks, that is not necessarily available in video surveillance. For comparison, our DTML methodology uses a classic CNN architecture set-up for metric learning. It does not require synthetic samples or any manual labeling.

To illustrate the effect of our dual-triplet loss function, we show the accuracy per training step in Figure 3.9 and 3.10. In both cases, using only the source term L_s does not significantly help the target camera to learn representations for the target camera. Also, as shown in Figure 3.10, using only the target term L_t may lead to decreases in performance as the training step continues due to overfitting. Having the whole dual-triplet ensure an increase in performance during training time. In Figure 3.9, we see that using both triplets gives the best performance in the long term. Note also that, the target term L_t would not achieve this performance without the initial help of the source fine-tuning at the beginning of the experiment. Indeed, that step helps reduce the domain gap and makes it easier to perform the pseudo-labeling.

To support our claims, Figure 3.11 shows TSNE projections of the impact of our proposed UDA method. Figure 3.11.a shows the projection of ten subjects in the evaluation set of the COX-S2V camera 2 viewpoint as if camera 2 was newly added to a surveillance network. These representations were generated using our feature extractor and fine-tuned using the source images

from the camera 3 viewpoint. The projections show that it is harder to classify subjects using only source representations. Figure 3.11.b shows again the projections of those ten subjects but with their representations fine-tuned with our proposed UDA method. Visually, there appears to be better discrimination between the representations of the different identities.

The proposed method was specifically designed to solve transfer problems where the domain shift between the source and target domains is small enough for the mutually-supervised training method. If the source and target embedding are not well-enough aligned, our mining approach will not be able to label correctly enough target pairs to be efficient. In this case, the mutually-supervised learning will not work properly. That being said, the method works with a moderate domain shift.

Methods	No. parameters	No. operations
Ours (Resnet50)	23M	3.8B
TBE-CNN (Ding & Tao, 2018)	46.4M	12.8B
HaarNet (Parchami *et al.*, 2017b)	13.1M	3.5B
CCM-CNN (Parchami *et al.*, 2017a)	2.4M	33.3M

Table 3.4 Comparison of model complexity for video surveillance.

Finally, the proposed method is intended to use with existing feature extractor architecture. The above experiment was made with an architecture ResNet50 which can be computationally heavy for real time applications. Table 3.4.5 show a comparison of the existing methods with the chosen architecture. It possible to try our methodology on less complex CNN architecture to achieve faster inference time, but it is important to note that CNN are for now computationally expensive for real-time applications. TBE-CNN (Ding & Tao, 2018) is a custom architecture of trunk and branch convolution layer that make it very computationally heavy. HaarNet (Parchami *et al.*, 2017b) use an ensemble of deep CNN also organise in trunk and branch architecture, but the CNNs as less parameters. CCM-CNN (Parchami *et al.*, 2017a) use a much more lightweight CNN architecture with cross correlation matching to achieve comparable results.

3.5 Conclusion

The triplet loss methodology give near state-of-the-art accuracy for the LFW evaluation set. We have successfully reproduced the results from the original paper on Facenet and have a ready-to-use face feature extractor for our Siamese network.

A feature extractor trained on public dataset do not work at best for video surveillance application. Indeed, video surveillance application are harder to solve. We see that there are a lot of potential improvement by fine-tuning the feature extractor on samples coming from the camera itself. It cannot be considered as a final solution since creating annotated samples for every surveillance that we require to install is very time expensive. Knowing there are huge improvement to gain with our current feature extractor architecture, we may explore non intrusive techniques that will help gain this improvement.

We propose a general method for domain adaptation of deep-distance metrics. The method is designed to resolve domain shift problems between the source and target domains. The source domain must have enough high-quality labeled samples to train a robust feature extractor, and the target domain can provide a small amount of unlabeled samples. Our method introduces dual-triplet loss and mutual-supervision learning. Dual triplets are the sum of triplets from the source and triplets from the target, which reduces model corruption after the domain adaptation process due to of the lack of target samples. The mutual-supervision process allows the formation of triplets from the unlabeled target samples using the knowledge of the source model. The method is used for a face recognition still-to-video application to provide unsupervised domain adaptation. Our method achieves results comparable to state-of-the-art and more complex methods, that are impractical because of the limitations of the camera use-case.

CHAPTER 4

A SIAMESE NETWORK FOR VIDEO DESCRIPTION

4.1 Introduction

Video description allows blind and visually impaired people to enjoy movies, TV series, and other video media. Manually performing description is tedious and time intensive. Movies and TV series generally come with video description for this audience, but as the internet is filled with more and more video content, cheaper video description becomes highly sought after. To this end, we seek to develop automatic video description that will allow people with visual problem to access video content.

Recently, some researchers have attempted to label data automatically by leveraging temporal and contextual information that can be obtained in videos like tracklets (Sharma *et al.*, 2019) (Wu *et al.*, 2013) (Cinbis *et al.*, 2011). It is important to note that these methods require an abundance of unlabeled data and the existence of co-occurring tracklets.

Face recognition for video, generally follows a specific process. First, the faces are segmented using a face detection algorithm. For each face patch, we produce a representation with a feature extractor . Each representation can be classified with a kNN-like classifier. We then apply this methodology to each frame so that we have all the face representations of the video. With a clustering algorithm, we group similar representations and give them their character name. The whole process is shown in Figure 4.1.

As the performance of detection and tracker techniques for faces improves, the task of identification becomes harder as the captures contain more occlusion, pose variation, and blur if the subjects are far away. To resolve this problem, a common and expensive solution is to create annotations with supervised learning on the identification model, which is not efficient. To respond to this challenge, researchers have developed self-supervised learning techniques in which the methodologies label the samples and then perform supervised learning. In this way, the

identification model is specialised for the video on which it performs video description. However, automatic labeling may not be perfect and can introduce labeling errors, and an insufficient number of samples will lead to overfitting. To automatically produce training samples from an unseen video, techniques use tracklets with the assumption that: samples within a same tracklet come from the same person and samples coming from co-occurring tracklets come from different people. In this way, it is possible to produce positive and negative samples from unlabeled data.

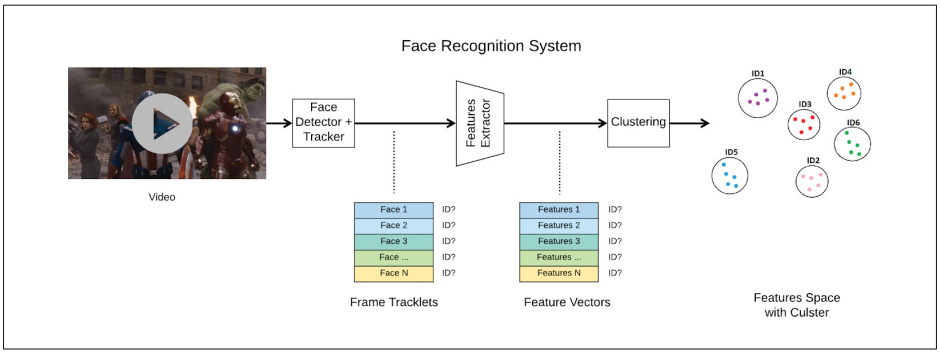


Figure 4.1 Face recognition system for video description .

4.2 Face Recognition for Video Description

In this section, we review the principal components of our face recognition system for video description. The components include a face segmentation technique, a face feature extractor trained with triplet loss on VGGface2, and feature clustering techniques.

4.2.1 Segmentation

Our face segmentation methodology is based on a face detector and tracking algorithm. At each frame, the detector captures faces with bounding box descriptors. On the first detection, it initiates a tracker that will follow the region of interest by updating its own representation of the region of interest. All samples captured by a tracker are packed together to form tracklets, which guarantees that samples within a tracklet come from the same person. In a case where the

detector do not detect a face with an existing tracker, the tracker will operate for 10 more frame. If there are no detection during those 10 frames, the last samples captured by the tracker will be rejected and the track will be terminated. This strategy will achieve a less fragmented track as the strategy will compensate for some false rejections from the detector. As a face detector, we use Multi-task Cascaded Convolutional Networks (MTCNN) (Zhang *et al.*, 2016) as it is a very well-known face detector that is robust and easy to use off-the-shelf. It is composed of three CNNs in a cascade, where the first CNN operates on the whole image and the others operate on the regions of interest given by the first one. For each, we set high detection thresholds [0.7, 0.9, 0.98] to minimize the false acceptance rate, to the detriment of the false rejection rate as the tracker algorithm should compensate for a given trajectory. We use Kernelized Correlation Filters (KCF) (Henriques, Caseiro, Martins & Batista, 2014) as a tracking algorithm as it is a general purpose tracking technique and is known to be robust and efficient.

4.2.2 Feature Extractor

Our feature extractor is based on a generally used DCNN architecture that is a ResNet50. The model is trained on the large-scale dataset VGGface2 with triplet loss, where this methodology learns a margin between classes in the Euclidean space. The training samples are resized to 160x160 with random horizontal flip. For a given face sample, the model produces a feature vector with a length of 128. The number of features may be small compared to that found in other research (256, 512), leading to a very small decrease in performance. However, it allows faster performance of clustering or projection techniques and is sufficient for our applications. The similarity of feature vectors can be directly measured by calculating the distance between them; small distances indicate that the samples come from the same persons and greater distances indicate that the samples come from different people.

4.2.3 Clustering

From the extracted feature vectors and the tracking information, we seek to automatically group classes. We will perform two types of clustering, one at the frame level and the other at the track

level, to visualize the performance of both methods. At the frame level, each representation is passed to the clustering algorithm. At track level, the mean representation of each tracklet is calculated and then passed to the clustering algorithm. The clustering techniques that we use in this experiment are Kmeans and agglomerative clustering as they are known clustering techniques and can easily operate in the Euclidean space. The more the feature extractor is invariant to changes in pose, the more the clustering accuracy should increase. For these techniques, the number of clusters must be given. To determine the number of clusters, we calculate the silhouette accuracy for each n cluster. The silhouette accuracy is a metric of the clustering quality between -1 and 1. -1 means that the samples are overlapping, 0 means that the samples are very close to the decision boundaries and 1 means that the clusters are perfectly defined from each other. The number of clusters is chosen from the n clusters that give the best clustering quality.

In this section, we review three clustering techniques: Kmeans, hierarchical agglomerative clustering (HAC), and spectral clustering. Each one has advantages and weaknesses, and it is necessary to understand them and to experiment them in order to determine the technique that will best serve our interest.

Kmeans

One of the classics and simplest clustering technique is the kmeans. It aims to partition n data of d dimensions into k clusters C. Each cluster is described by is centroids, which are the means of the data within the cluster. The algorithm will seek to minimize the within-cluster sum-of-square (inertia) criterion:

$$MSE = \sum_{i=0}^{n} \min_{\mu_j \in C} \left(\left\| x_i - \mu_j \right\|^2 \right) \tag{4.1}$$

The kmeans does not work well for data distributions that are not convex and isotropic. Also, since the inertia is not a normalized metric, the Euclidean distance may become inflated with data of very high dimensionality.

Hierarchical Agglomerative Clustering

Hierarchical agglomerative clustering (HAC) is a metric distance-based clustering technique that is known to be computationally expensive. It consists of repeatedly combining the two nearest clusters until the desired number of clusters is reached. The combination is made with a linkage criteria.

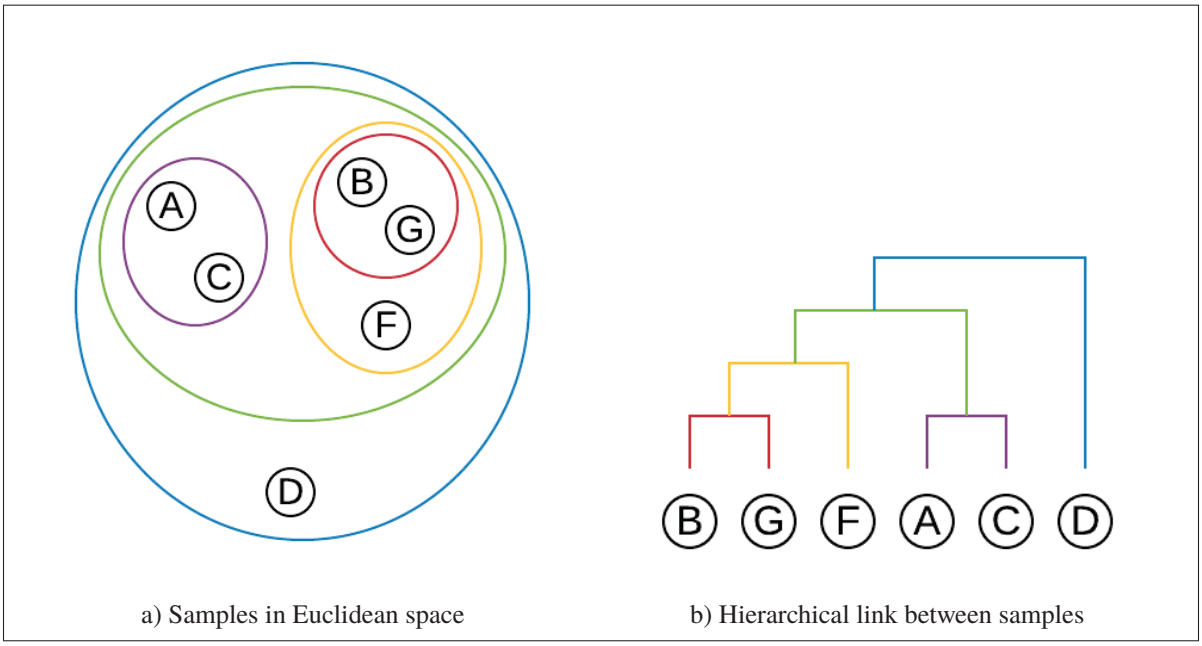


Figure 4.2 Example of hierarchical agglomerative clustering

By default, we will use the Euclidean distance with a ward linkage criteria for the experiments. The ward linkage criteria aims to minimize the total variance within the clusters.

4.3 Video Description with Pre-Trained CNN Backbone

We first test the ability of our feature extractor trained on a public dataset to cluster the different identities in the first episode of the TV series Big Bang Theory (BBT). The objective is to group the subject's samples in an unsupervised manner. The number of clusters is determined by computing the silhouette score for a number of possible clusters and selecting the number that maximizes this score. To compare our result with that of other researchers, we take the

annotations from CVPR 2013, which contain bounding boxes and identity labels. We use their bounding boxes to capture the region of interest. From the ROI, we extract the features, apply the clustering technique, and compute our different performance metrics with the identity label from CVPR 2013.

The experiment is done in two parts. The first part uses only the first scene and the second part uses the entire episode. For the second part, we examine the performance gain from computing the average features of tracklets. Using only the first scene gave very bad results as there were not enough samples to form an identity group easily. Thus, the results for the tracklet average are not shown for the first scene.

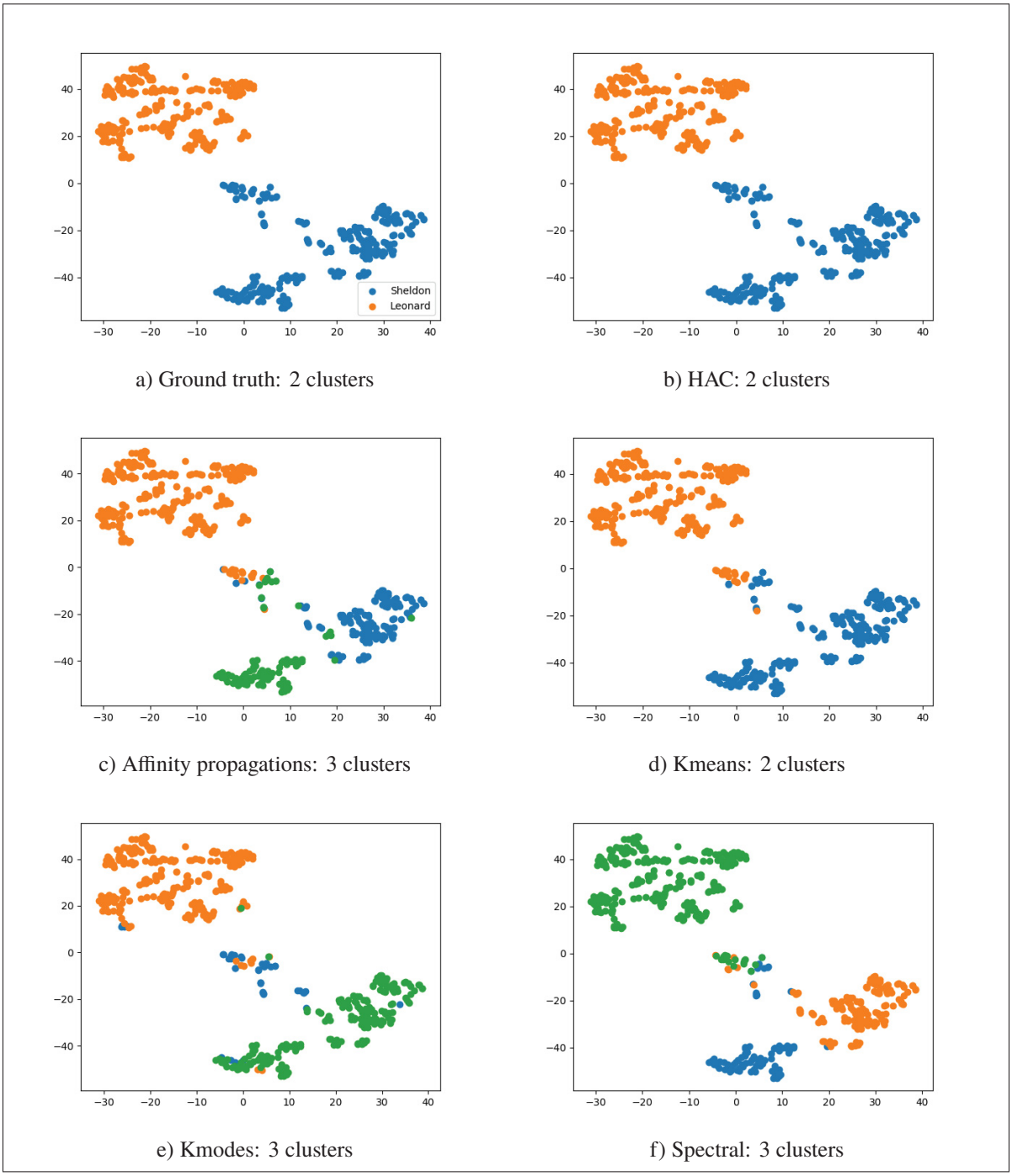
4.3.1 Experiment on Small Video Segment

Before doing the experiment using the entire episode, we test our framework on the first scene. It allows us to clearly see the identities cluster with TSNE projection and to visualize the effectiveness of the clustering algorithms. The first scene is captured by taking the first 500 frames. With TSNE, we project all of the representations onto a 2-dimensional plan. The grountruth labels are shown with the result of the clustering techniques and the predicted number of clusters.

Table 4.1 Clustering results on the first 500 frames of BBT episode 1.

Clustering technique	Weighted purity	V measure	Silhouette score	Predicted number of clusters
HAC	1	1	0.1961	2
Kmeans	0.9626	0.8061	0.1984	2
Spectral	0.9648	0.6596	-	3
Kmodes	0.9670	0.6871	0.1968	3
Affinity propagation	0.9648	0.6593	0.1950	3

For this reduced case, simple clustering techniques such as Kmeans and HAC perform best. With this feature extractor, kernel base clustering may be efficient enough to group tracklets and not group identities. We observe that the silhouette score for clustering techniques are



a) Ground truth: 2 clusters

b) HAC: 2 clusters

c) Affinity propagations: 3 clusters

d) Kmeans: 2 clusters

e) Kmodes: 3 clusters

f) Spectral: 3 clusters

Figure 4.3 TSNE projections of BBT episode 1 reduced.

approximately the same, which means that setting apart the groundtruth, the clustering quality of each technique is approximately the same without being correct.

The weighted purity acts as expected; it considers the homogeneity of the clustering, but does not penalize for incorrect compactness. It is unlikely that we will use this metric to compare the clustering fairly. On the other hand, the V measure seems to represent the clustering performance, penalizing the score when the number of cluster is incorrect.

4.3.2 Experiment on Large Video Segment

In this part, we experiment on the entire episode 1 of BBT. We first try to cluster all of the face samples with the current techniques which we call frame-level clustering. Then, we compute the mean representation of each track and again perform the same clustering techniques as track-level clustering. The objective is to cluster the 7 present identities successfully.

Frame Level Clustering

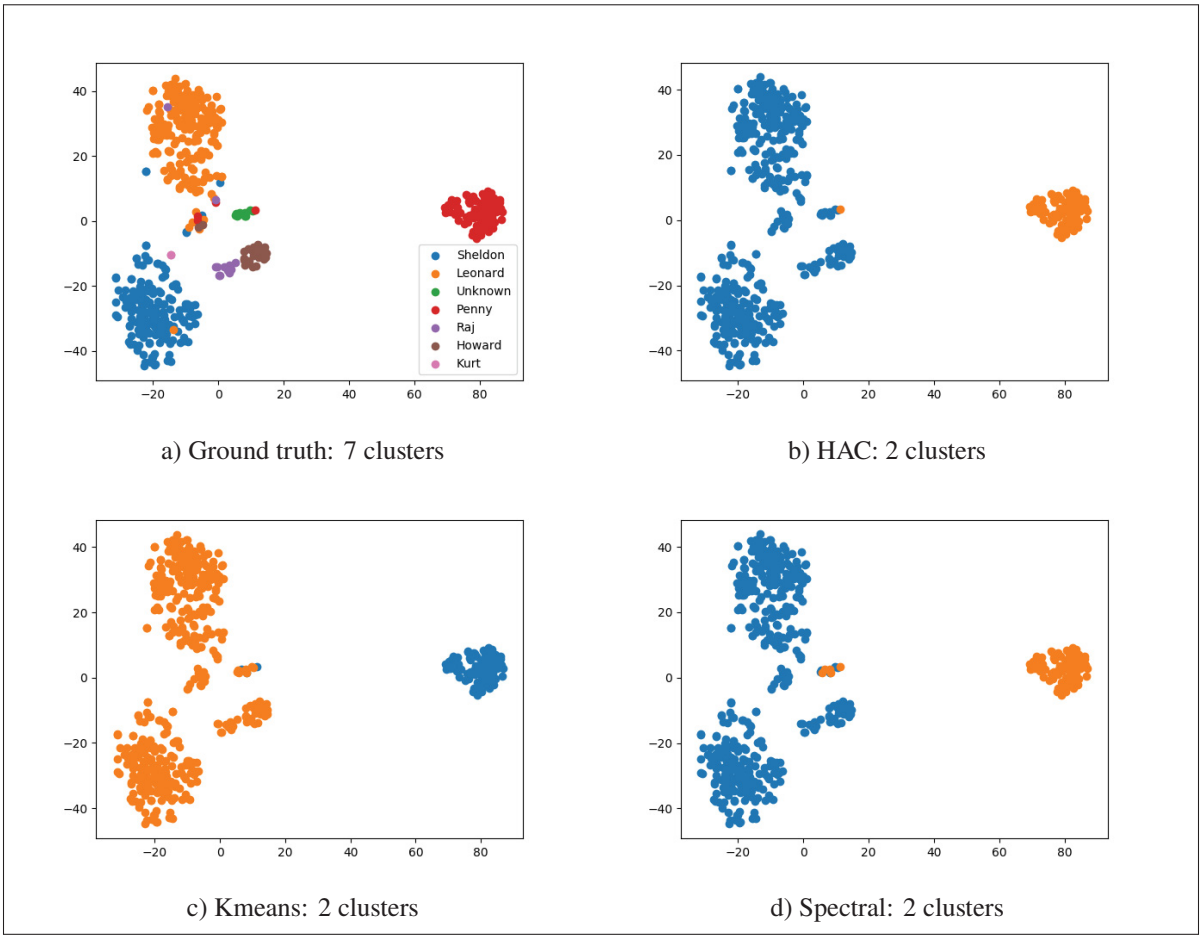
In this section, we apply the clustering techniques to all of the face representations. To visualize the projections, we have randomly selected 500 of the 39183 captured samples with the CVPR 2013 bounding box annotations. The TSNE projection takes a lot of time when there are more than 500 samples.

It was not possible to experiment using the Kmodes and Affinity propagation clustering techniques as they require significant computing time.

Table 4.2 Clustering results on BBT episode 1 at frame level.

Clustering technique	Weighted purity	V measure	Predicted number of clusters
HAC	0.5323	0.4370	2
Kmeans	0.5314	0.4290	2
Spectral	0.5315	0.4296	2

The combination of clustering with silhouette scores to find the number of clusters automatically does not appear to be the best solution in this case. Even with a better methodology for determining the number of clusters, it is still a challenge to cluster each identity correctly.



a) Ground truth: 7 clusters

b) HAC: 2 clusters

c) Kmeans: 2 clusters

d) Spectral: 2 clusters

Figure 4.4 Frame level TSNE projections of BBT episode 1 with predicted clustering.

Table 4.3 Clustering results on BBT episode 1 at frame level per number of clusters.

	Kmeans		HAC	
Number of clusters	Weighted purity	V measure	Weighted purity	V measure
2	0.5314	0.4290	0.5323	0.4370
3	0.8481	0.6872	0.8672	0.7641
4	0.8897	0.7974	0.9151	0.8653
5	0.8912	0.7451	0.9151	0.8127
6	0.9312	0.7702	0.9581	0.8403
7	0.9467	0.7505	0.9739	0.8542
8	0.9487	0.7310	0.9739	0.7992

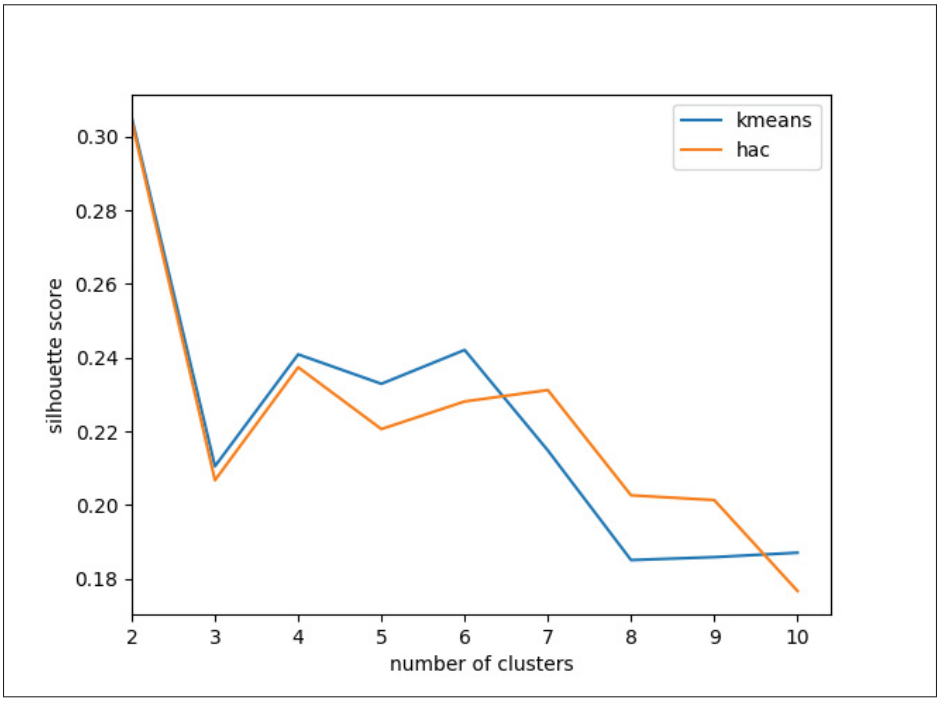


Figure 4.5 Frame level silouhette score per k clusters.

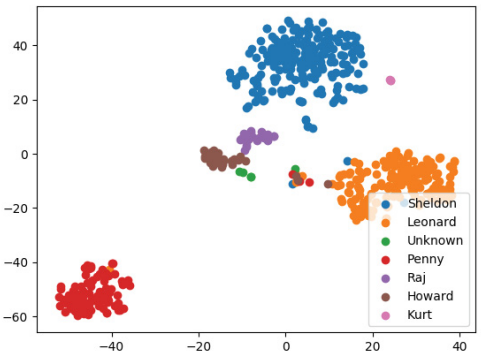
Track Level Clustering

In this experiment, we group the representations of a track into a single representation by computing the mean. This gives slightly better results and is also faster to cluster.

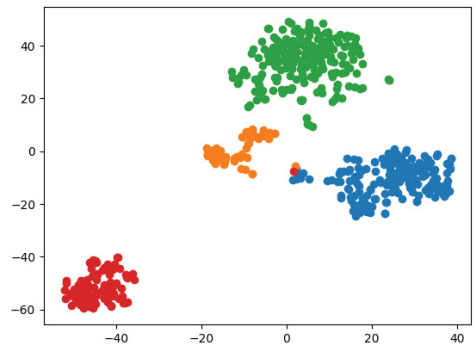
Table 4.4 Clustering results on BBT episode 1 at track level.

Clustering technique	Weighted purity	V measure	Silhouette score	Predicted number of clusters
HAC	0.5531	0.4799	0.2706	2
Kmeans	0.5531	0.4799	0.2710	2
Spectral	0.5531	0.4681	-	2
Kmodes	0.5531	0.4695	0.2668	2
Affinity propagation	0.9260	0.8595	0.3051	4

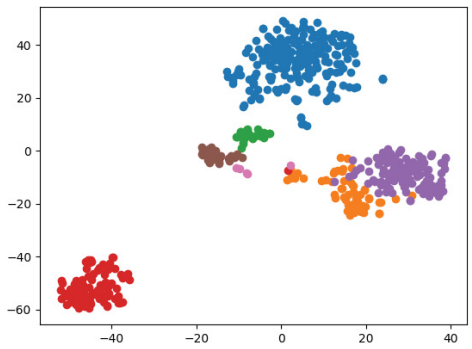
The track-level clusters visualization appears to be a bit more defined than frame-level visualization. If we compare the V measure of the Kmeans clustering, the frame-level clustering score is 0.4290 while the track-level clustering score is 0.4799. This is a small improvement,



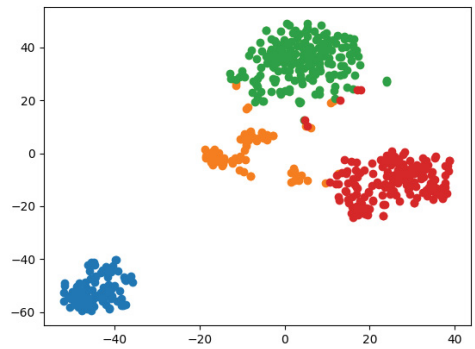
a) Ground truth: 7 clusters



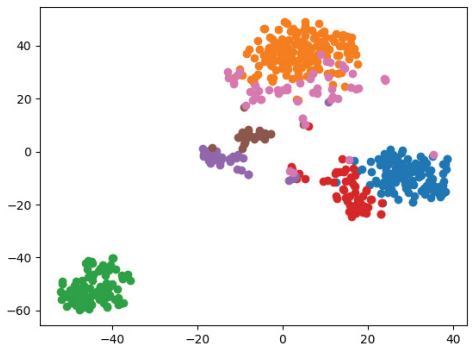
b) HAC: 4 clusters



c) HAC: 7 clusters

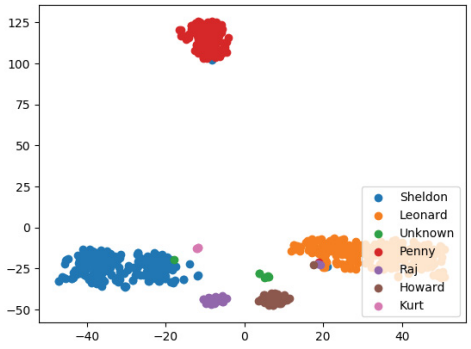


d) Kmeans: 4 clusters

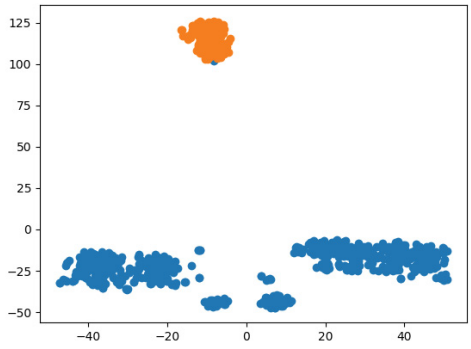


e) Kmeans: 7 clusters

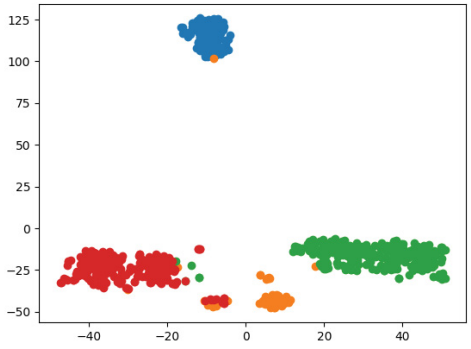
Figure 4.6 Frame level TSNE projections of BBT episode 1 with selected number of clusters.



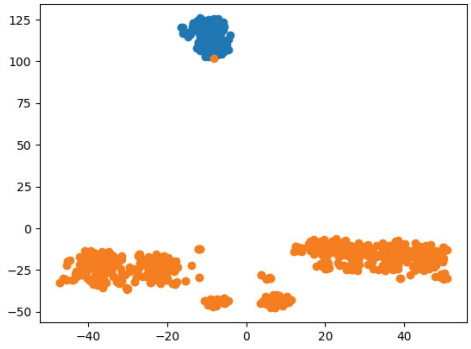
a) Ground truth: 7 clusters



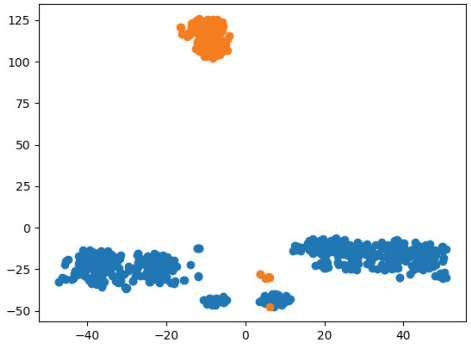
b) HAC: 2 clusters



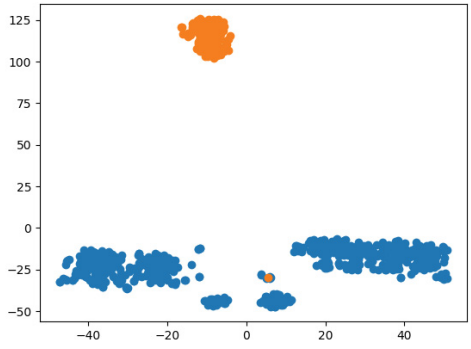
c) Affinity propagations: 4 clusters



d) Kmeans: 2 clusters



e) Kmodes: 2 clusters



f) Spectral: 2 clusters

Figure 4.7 TSNE projection of BBT episode 1 at track level.

but it still does not cluster the characters correctly. However, the affinity propagation technique seems to work better than the other techniques, and by forcing the number of clusters on the

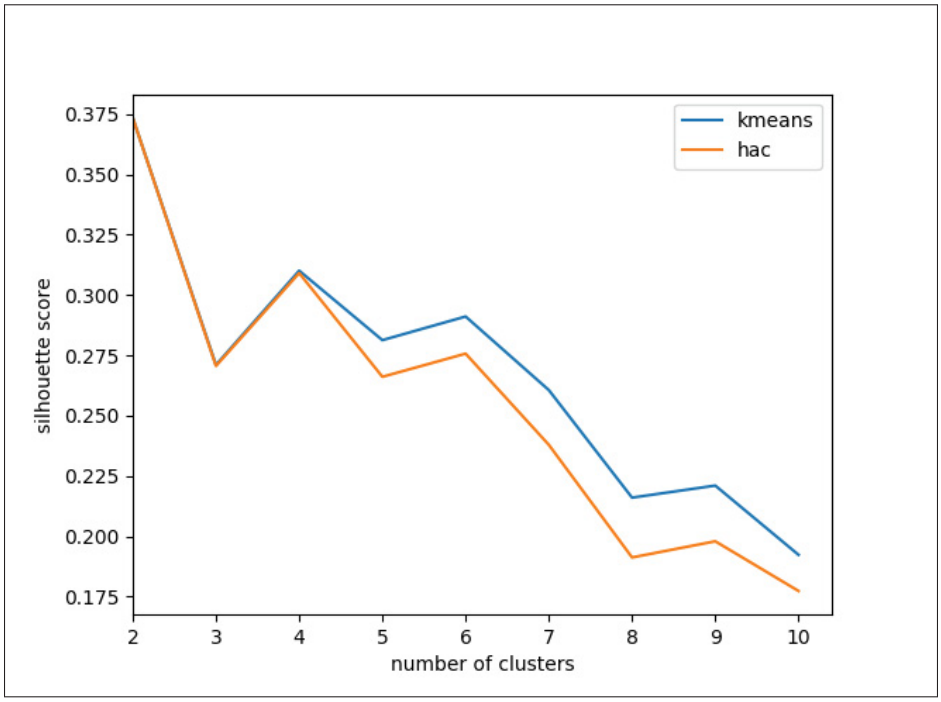


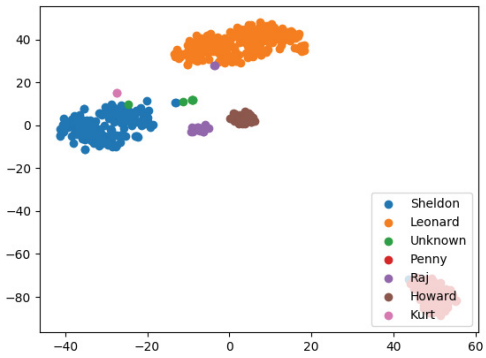
Figure 4.8 Track level silouhette score per k clusters.

basic clustering technique, we observe that we obtain better results with the Kmeans and HAC techniques.

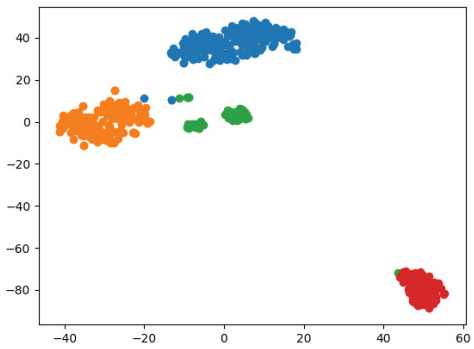
Table 4.5 Clustering results on BBT episode 1 at track level per number of clusters.

	Kmeans		HAC	
Number of clusters	Weighted purity	V measure	Weighted purity	V measure
2	0.5531	0.4799	0.5531	0.4799
3	0.8778	0.8097	0.8730	0.7987
4	0.9212	**0.8736**	0.9309	**0.8967**
5	0.9292	0.8172	0.9309	0.8207
6	0.9646	0.8383	0.9678	0.8478
7	0.9727	0.8115	0.9678	0.7942
8	0.9711	0.7755	0.9678	0.7668

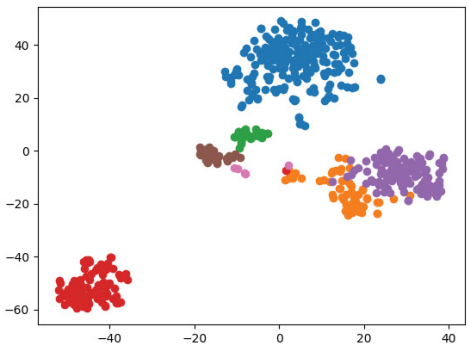
As observed in the frame-level experiment, we obtain greater performance when using a better solution than silhouette score to select the number of clusters. Selecting 4 clusters gives the best results for the Kmeans and HAC techniques for V measures of, respectively, 0.8736 and



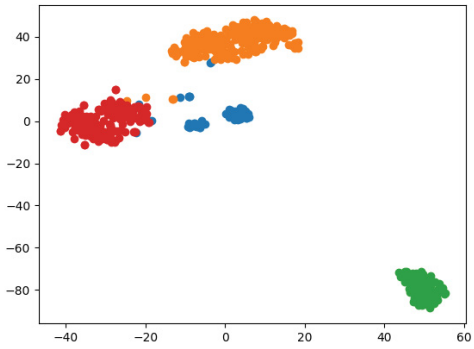
a) Ground truth: 7 clusters



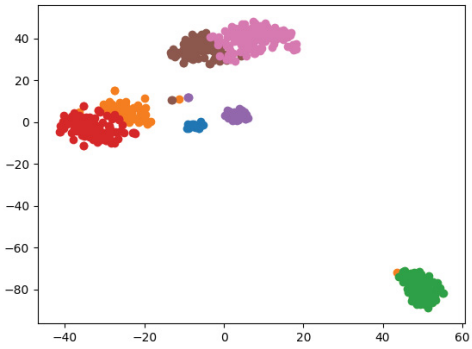
b) HAC: 4 clusters



c) HAC: 7 clusters



d) Kmeans: 4 clusters



e) Kmeans: 7 clusters

Figure 4.9 Track level TSNE projections of BBT episode 1 with selected number of clusters.

0.8967, which is better than the affinity propagation at 0.8595. We observe that track-level clustering works better and more quickly than frame-level clustering if there are many samples. Using Kmeans with 4 clusters, the frame-level and track-level clustering give V measures of, respectively, 0.7974 and 0.8736. Unfortunately, the system is still not able to clusters all of the 7 characters.

4.4 Face Clustering Experiment with Domain Adaptation Based on Tracklets

Exploiting the tracklets information can produce reliably trainable samples. A tracklet is made of a series of regions of interest captured during consecutive frames. The tracking algorithm attempts to retrieve from each frame approximately the same pattern as the previous frame.

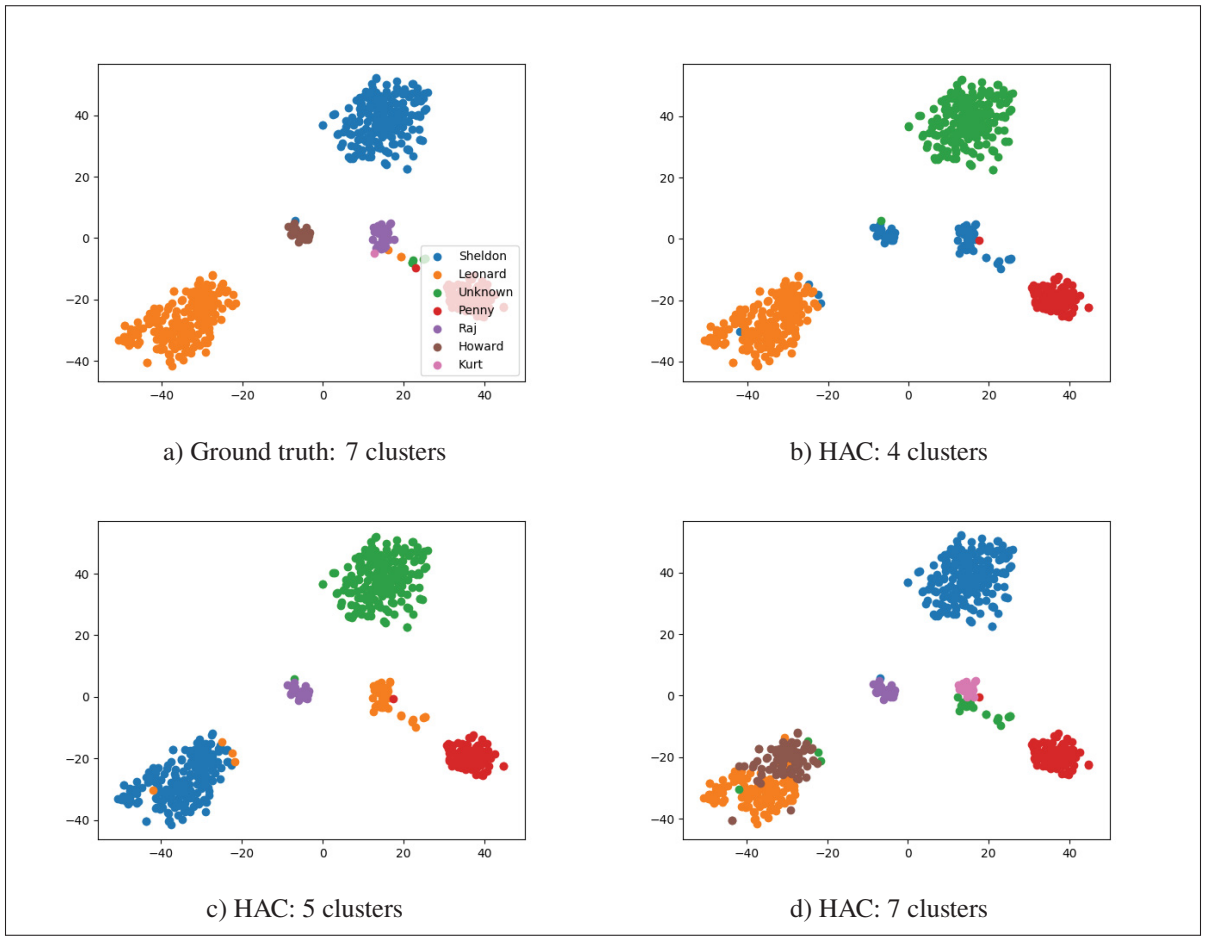
We pose the hypothesis that samples within a tracklet are from the same person and samples from co-occurring tracklets come from different people. For the BBT series, it is a fair hypothesis as these conditions will always be true compared to other shows. Thus, samples within a tracklet will be used to form positive pairs and samples from co-occurring tracklets will be used to obtain negative pairs to form trainable triplets. However, this method does not work with singleton track. For each training batch, we leverage an equal number of samples from each co-occurring tracklets, apply the semi-hard mining strategy, and train the FE with the formed triplets.

To augment the quality of the generated representations from the TV series, we fine-tune the model with samples from the video itself. For triplet loss to work, it is necessary to be able to form trainable triplets. We create these triplets by taking positive pairs from samples within the same track and by taking negative pairs from samples from co-occurring tracks.

We evaluate our methodology using only HAC as it was the technique that gave the best results in the previous experiment. Also, we did not select the number of clusters with the silhouette score since it did not give good results.

Frame Level Clustering

Fine tuning the model using the tracklets improves the clustering performance. The character representations seems to be a bit more defined by looking at the projections. Also, the V measure is already better than that of the previous experiment with a score of 0.9014 with 5 clusters (previously it was 0.8967 with 4 clusters). Using the silhouette score to predict the number of clusters, we would get 4 clusters, which is a major improvement compared to the previous experiment.



a) Ground truth: 7 clusters

b) HAC: 4 clusters

c) HAC: 5 clusters

d) HAC: 7 clusters

Figure 4.10 Frame level TSNE projections of BBT episode 1 with tracklets fine-tuned.

Table 4.6 HAC results on BBT episode 1 at frame level with tracklets fine-tuned.

Number of cluster	V measure	Silhouette score
2	0.5611	0.2921
3	0.8006	0.3457
4	0.8726	**0.3668**
5	**0.9014**	0.3666
6	0.8352	0.2653
7	0.8405	0.2597
8	0.7849	0.1535

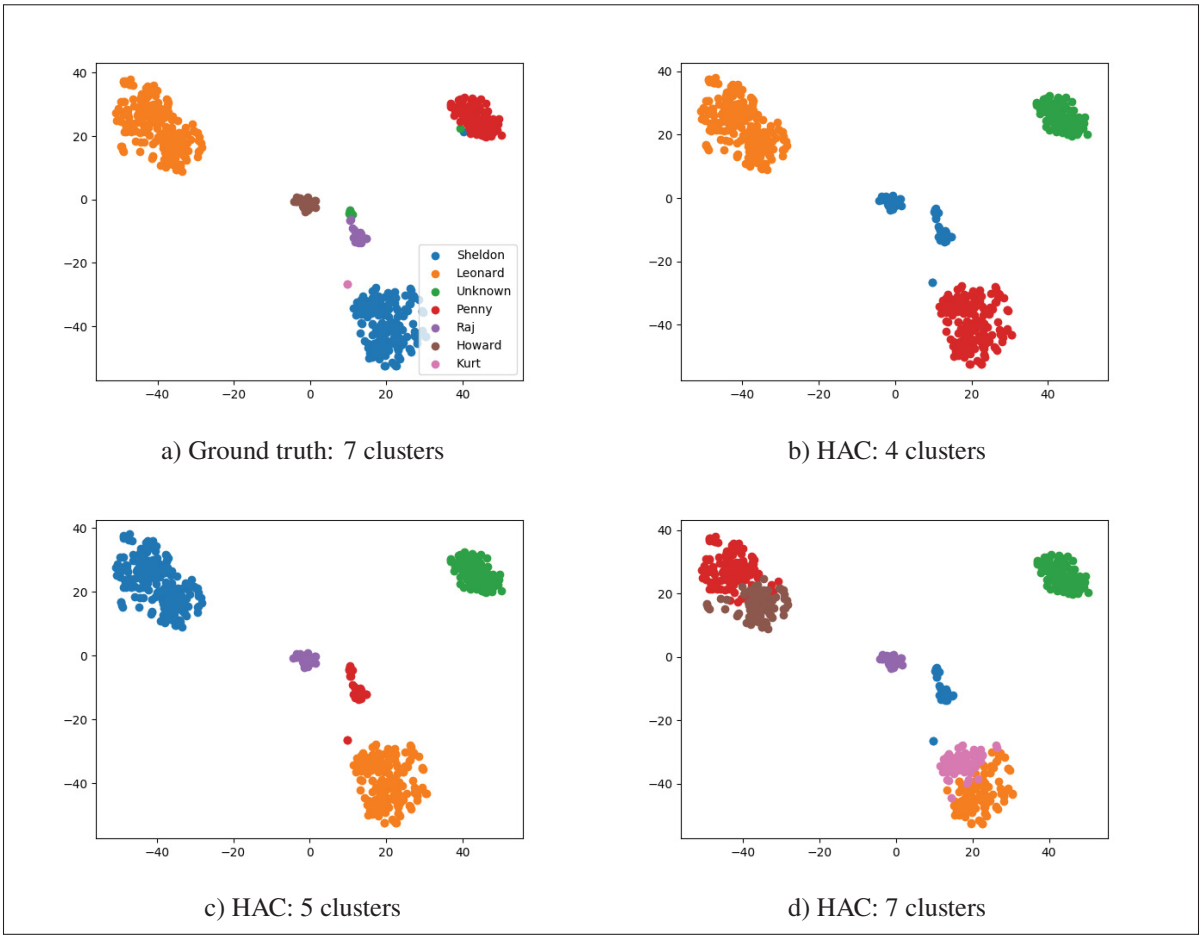
Track Level Clustering

We compute the mean of each set of representations in a tracklet, to perform clustering. The results are shown in Table 4.7. As previously mentioned, this methodology can be computed more quickly and gives better results if there are many tracks. Compared to the frame-level clustering, it improves the V measure at 0.9712 over 0.9014 which is a significant improvement. Also, in this case, the silhouette score would have given the best number of clusters.

Table 4.7 HAC results on BBT episode 1 at track level with tracklets fine-tuned.

Number of cluster	V measure	Silhouette score
2	0.6359	0.3604
3	0.8301	0.4477
4	0.9403	0.4704
5	**0.9712**	**0.4722**
6	0.8893	0.3379
7	0.8314	0.2223
8	0.8381	0.2229

However, we observe in Table 4.7 that we are still not able to correctly group the seven identities, even if we force the number of clusters to the exact number of identities. The two last identities to be grouped has very few representations and the clustering technique is splitting the big clusters of representations rather than grouping the very small and overlapping clusters as we see in Figure 4.4.



a) Ground truth: 7 clusters

b) HAC: 4 clusters

c) HAC: 5 clusters

d) HAC: 7 clusters

Figure 4.11 Track level TSNE projections of BBT episode 1 with tracklets fine-tuned.

Table 4.8 HAC with 5 clusters confusion matrix on BBT episode 1 at track level with tracklets fine-tuned.

	Predicted Class						
Ground Truth	Sheldon	Leonard	Penny	Howard	Raj	Kurt	Unknown
Sheldon	**202**	0	1	0	0	0	0
Leonard	0	**236**	0	0	0	0	0
Penny	0	0	**109**	0	0	0	0
Howard	0	0	0	**37**	0	0	0
Raj	0	0	0	0	**25**	0	0
Kurt	0	0	0	0	4	**0**	0
Unknown	0	0	1	0	7	0	**0**

In Table 4.8, we show the resulting confusion matrix for clustering our identities with five groups, which give the highest V measure. We may achieve high clustering accuracy, but the matrix is clearly showing that our methodology is not able to regroup identities that have few apparitions.

4.5 Conclusion

Fine-tuning the feature extractor in the self-supervised manner gives a better performance for the clustering technique. Unfortunately, the model is not able to discriminate the characters that appear less frequently, like the characters Kurt and Unknown. Generally, for a long video, reducing the frame features to track features seems to give better results of clustering. The bad frame features in a track may be smoothed by the others good face samples within the same track.

The silhouette score methodology generally works for certain scenarios. At the frame level, the methodology gives a number of clusters of four (Table 4.6), which was not the best possible number of clusters to maximize the V measure. However, at the track level, the methodology gives the best number of possible clusters. We observe that track level samples produce more distinct clusters.

We show in Table 4.9 the performance gain of a fine-tuned model, compared to our Facenet base model. From the base model, we can observe clear performance gains by using track level samples rather than frame level samples within a reasonable number of clusters (below seven). The more the number of clusters approaches the real number of separable clusters, the less gain there is. The fine-tuned model at the frame level outperforms the base model by a margin of 8% for five clusters, which already shows the gain obtained using our methodology. However, the combination of fine tuning the model and using the track level gave our best V measure of 97 % for five clusters. Even if the real number of clusters is seven, the majority of the identities have been correctly regrouped.

Table 4.9 HAC clustering V measure on BBT episode 1 comparison.

	Base model		Tracklet fine-tuned model	
Number of clusters	Frame level	Track level	Frame level	Track level
2	0.4370	0.4799	0.5611	0.6359
3	0.7641	0.7987	0.8006	0.8301
4	0.8653	0.8967	0.8726	0.9403
5	0.8127	0.8207	0.9014	**0.9712**
6	0.8403	0.8478	0.8352	0.8893
7	0.8542	0.7942	0.8405	0.8314
8	0.7992	0.7668	0.7849	0.8381

CONCLUSION AND RECOMMENDATIONS

The face recognition field has made a lot of progress. Face segmentation techniques capture more and more faces under difficult conditions that make the classification task more challenging as regions-of-interest suffer more from issues such as pose variation, occlusion, and differences in illumination. Also, in this research, we seek face recognition for video applications and, more precisely, video surveillance and video description. Robust feature extractors are trained with still face images that are not adapted for video applications. It makes two kind of image domains: the still domain and video domain. In our research, we attempt to close the gap between these two domains. Training with still images improves the performance of models as there is an abundance of labeled data, but there are not as many video samples. Specifically, we seek to enhance the performance of models on previously unseen video material where manually labeling data is expensive.

In Chapter 2, we reviewed the experimental datasets and metrics used for this research. We used Vggface2 to learn our first base face representations and LFW to compare the methodology to the literature. For the video surveillance application, we used the COX-S2V dataset, which contains video samples from 1000 subjects. For video description, we used the first episode of the TV series The Big Bang Theory with the annotations from CVPR 2013.

In Chapter 3, we applied a Siamese network for video surveillance. We trained a base feature extractor on the still face images of the VGGface2 dataset with triplet loss methodology to be used in further experiments. With this base FE, we attempted to perform domain adaptation from still to video with the COX-S2V dataset. We employed multiple methodologies to attempt to close the domain gap between video and still images such as supervised learning with annotated samples from the target domain. Doing this gave the best performance, but it is unpractical in real-world applications as creating these annotated samples is expensive. To address this problem, we introduced DTML with mutual supervision to automatically calibrate newly installed video

sources without the need for annotated samples or a large amount of samples, making it very practical for real world applications. This mythology significantly enhanced the quality of the representations produced by the feature extractor from new domains but did not surpass classic supervised calibration. It may not work on target video if the domain gap is too large, as the mutual supervision method is barely able to label the target samples.

In Chapter 4, we addressed the problem of video description, for which the problem is to regroup identities in movies or videos. The experiment was performed on the first episode of TV series The Big Bang Theory, from which we attempted to regroup seven characters. The feature extractor trained with a large set of still images did not perform well at regrouping the characters as the face samples captured by the robust face detector MTCNN suffered a lot from occlusion, changes in illumination, and expression changes. We experimented with our base feature extractor and certain clustering techniques to regroup the identities for the whole episode. Clustering was attempted at two levels: the frame level and the tracking level. The tracking level clustering provided better clustering accuracy and greater speed in all cases. However, the base feature extractor did not produce optimal results. To close again the domain gap, we attempted to leverage the spatio-temporal information given by tracklets. We can label sample pairs as being the same or not the same with the following hypothesis: samples within a track are from the same identity and samples from co-occurring tracks are from different identities. With these labels, we formed triplets and fine-tuned the feature extractor with classic triplet loss.

Future Work

Closing the gap between image domains still remains a challenge. With our work, one may attempt to combine the mutual supervision method from Chapter 3 with the tracklets information experimented with in Chapter 4. This combination should work for video surveillance and video description applications where it is necessary to avoid manually labeling data.

REFERENCES

Bengio, Y., Courville, A. & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8), 1798–1828.

Bianco, S., Cadene, R., Celona, L. & Napoletano, P. (2018). Benchmark analysis of representative deep neural network architectures. *IEEE Access*, 6, 64270–64277.

Cinbis, R. G., Verbeek, J. & Schmid, C. (2011, Nov). Unsupervised metric learning for face identification in TV video. *2011 International Conference on Computer Vision*, pp. 1559-1566. doi: 10.1109/ICCV.2011.6126415.

Comaschi, F., Stuijk, S., Basten, T. & Corporaal, H. (2015). Online multi-face detection and tracking using detector confidence and structured SVMs. *2015 12th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pp. 1–6.

Ding, C. & Tao, D. (2018). Trunk-branch ensemble convolutional neural networks for video-based face recognition. *IEEE transactions on pattern analysis and machine intelligence*, 40(4), 1002–1014.

Ekladious, G., Lemoine, H., Granger, E., Kamali, K. & Moudache, S. (2020). Dual-triplet metric learning for unsupervised domain adaptation in video-based face recognition. *arXiv preprint arXiv:2002.04206*.

Ganin, Y. & Lempitsky, V. (2014). Unsupervised Domain Adaptation by Backpropagation. *arXiv*, stat.ML(1409.7495v2).

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. & Bengio, Y. (2014). Generative adversarial networks. *arXiv preprint arXiv:1406.2661*.

Hadsell, R., Chopra, S. & LeCun, Y. (2006). Dimensionality reduction by learning an invariant mapping. *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, 2, 1735–1742.

He, K., Zhang, X., Ren, S. & Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778.

Henriques, J. F., Caseiro, R., Martins, P. & Batista, J. (2014). High-speed tracking with kernelized correlation filters. *IEEE transactions on pattern analysis and machine intelligence*, 37(3), 583–596.

Hoffer, E. & Ailon, N. (2015). Deep Metric Learning Using Triplet Network. *Similarity-Based Pattern Recognition*, pp. 84–92.

Hong, S., Im, W., Ryu, J. & Yang, H. S. (2017). SSPP-DAN- Deep Domain Adaptation Network for Face Recognition with Single Sample Per Person. *arXiv*, cs.CV(1702.04069v4).

Hsu, R.-L. (2002). *Face detection and modeling for recognition*. Michigan State University. Department of Computer Science & Engineering.

Huang, Z., Shan, S., Wang, R., Lao, S., Kuerban, A. & Chen, X. (2015). A benchmark and comparative study of video-based face recognition on COX face database. *IEEE Transactions on Image Processing*, 14(12), 5967–5981.

Jain, A. K. & Li, S. Z. (2011). *Handbook of face recognition*. Springer.

Jain, V. & Learned-Miller, E. (2010). *Fddb: A benchmark for face detection in unconstrained settings*.

Krizhevsky, A., Sutskever, I. & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, pp. 1097–1105.

Laradji, I. H. & Babanezhad, R. (2018). M-ADDA: Unsupervised Domain Adaptation with Deep Metric Learning. *CoRR*, abs/1807.02552.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y. & Berg, A. C. (2016). Ssd: Single shot multibox detector. *European conference on computer vision*, pp. 21–37.

Luo, Z., Hu, J., Deng, W. & Shen, H. (2018, May). Deep Unsupervised Domain Adaptation for Face Recognition. *2018 13th IEEE International Conference on Automatic Face Gesture Recognition (FG 2018)*, pp. 453-457. doi: 10.1109/FG.2018.00073.

Mokhayeri, F., Granger, E. & Bilodeaun, G. (2019). Domain-Specific Face Synthesis for Video Face Recognition from a Single Sample Per Person. *IEEE Transactions on Information Forensics and Security*, 14(3).

Parchami, M., Bashbaghi, S. & Granger, E. (2017a). CNNs with cross-correlation matching for face recognition in video surveillance using a single training sample per person. *Proceedings of the 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pp. 1–6.

Parchami, M., Bashbaghi, S. & Granger, E. (2017b). Video-based face recognition using ensemble of haar-like deep convolutional neural networks. *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, pp. 4625–4632.

Parchami, M., Bashbaghi, S., Granger, E. & Sayed, S. (2017c). Using deep autoencoders to learn robust domain-invariant representations for still-to-video face recognition. *Proceedings of the 14th IEEE International Conference on dvanced Video and Signal Based Surveillance (AVSS)*.

Parkhi, O. M., Vedaldi, A. & Zisserman, A. (2015). Deep face recognition.

Q. Cao, Q., Shen, L., Xie, W., Parkhi, O. M. & Zisserman, A. (2018). VGGFace2: A dataset for recognising face across pose and age. *2018 International Conference on Automatic Face and Gesture Recognition*.

Redmon, J., Divvala, S., Girshick, R. & Farhadi, A. (2016). You only look once: Unified, real-time object detection. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788.

Ren, S., He, K., Girshick, R. & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, pp. 91–99.

Rosenberg, A. & Hirschberg, J. (2007). V-measure: A conditional entropy-based external cluster evaluation measure. *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, pp. 410–420.

Rousseeuw, P. J. (1987). Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20, 53–65.

Schroff, F., Kalenichenko, D. & Philbin, J. (2015a). Facenet: A unified embedding for face recognition and clustering. *Proceedings of the Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pp. 815-823.

Schroff, F., Kalenichenko, D. & Philbin, J. (2015b). Facenet: A unified embedding for face recognition and clustering. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 815–823.

Sharma, V., Tapaswi, M., Sarfraz, M. S. & Stiefelhagen, R. (2019). Self-Supervised Learning of Face Representations for Video Face Clustering. *2019 14th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2019)*, 1-8.

Sohn, K., Liu, S., Zhong, G., Yu, X., Yang, M.-H. & Chandraker, M. (2017). Unsupervised Domain Adaptation for Face Recognition in Unlabeled Videos. *arXiv*, cs.CV, cs.AI(1708.02191v1).

Sohn, K., Shang, W., Yu, X. & Chandraker, M. (2019). Unsupervised Domain Adaptation for Distance Metric Learning. *International Conference on Learning Representations*.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. & Rabinovich, A. (2015). Going deeper with convolutions. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9.

Tapaswi, M., Bäuml, M. & Stiefelhagen, R. (2012). "Knock! Knock! Who is it?" probabilistic person identification in TV-series. *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2658–2665.

Tapaswi, M., Parkhi, O. M., Rahtu, E., Sommerlade, E., Stiefelhagen, R. & Zisserman, A. (2014). Total cluster: A person agnostic clustering method for broadcast videos. *Proceedings of the 2014 Indian Conference on Computer Vision Graphics and Image Processing*, pp. 1–8.

Viola, P. & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*, 1, I–I.

Wen, G., Chen, H., Cai, D. & He, X. (2018). Improving face recognition with domain adaptation. *Neurocomputing*, 287, 45 - 51. doi: https://doi.org/10.1016/j.neucom.2018.01.079.

Wen, Y., Zhang, K., Li, Z. & Qiao, Y. (2016). A discriminative feature learning approach for deep face recognition. *European conference on computer vision*, pp. 499–515.

Wolf, L., Hassner, T. & Maoz, I. (2011, June). Face recognition in unconstrained videos with matched background similarity. *CVPR 2011*, pp. 529-534. doi: 10.1109/CVPR.2011.5995566.

Wu, B., Lyu, S., Hu, B. & Ji, Q. (2013, Dec). Simultaneous Clustering and Tracklet Linking for Multi-face Tracking in Videos. *2013 IEEE International Conference on Computer Vision*, pp. 2856-2863. doi: 10.1109/ICCV.2013.355.

Yang, S., Luo, P., Loy, C.-C. & Tang, X. (2016). Wider face: A face detection benchmark. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5525–5533.

Zhang, Z., Luo, P., Loy, C. C. & Tang, X. (2016). Joint face representation adaptation and clustering in videos. *European conference on computer vision*, pp. 236–251.