Karlsruher Institut für Technologie

Self-supervised Face Representation Learning

zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

von der KIT-Fakultät für Informatik
des Karlsruher Instituts für Technologie (KIT)

genehmigte

Dissertation
von

Vivek Sharma

aus Siliguri, India

Tag der mündlichen Prüfung:	14. May 2020
Hauptreferent:	Prof. Dr.-Ing. Rainer Stiefelhagen Karlsruher Institut für Technologie
Korreferent:	Prof. Dr.-Ing. Tamim Asfour Karlsruher Institut für Technologie

Abstract

This thesis investigates fine-tuning deep face features in a self-supervised manner for discriminative face representation learning, wherein we develop methods to automatically generate pseudo-labels for training a neural network. Most importantly solving this problem helps us to advance the state-of-the-art in representation learning and can be beneficial to a variety of practical downstream tasks. Fortunately, there is a vast amount of videos on the internet that can be used by machines to learn an effective representation. We present methods that can learn a strong face representation from large-scale data be the form of images or video.

However, while learning a good representation using a deep learning algorithm requires a large-scale dataset with manually curated labels, we propose self-supervised approaches to generate pseudo-labels utilizing the temporal structure of the video data and similarity constraints to get supervision from the data itself.

We aim to learn a representation that exhibits small distances between samples from the same person, and large inter-person distances in feature space. Using metric learning one could achieve that as it is comprised of a pull-term, pulling data points from the same class closer, and a push-term, pushing data points from a different class further away. Metric learning for improving feature quality is useful but requires some form of external supervision to provide labels for the same or different pairs. In the case of face clustering in TV series, we may obtain this supervision from tracks and other cues. The tracking acts as a form of high precision clustering (grouping detections within a shot) and is used to automatically generate positive and negative pairs of face images. Inspired from that we propose two variants of discriminative approaches: Track-supervised Siamese network (TSiam) and Self-supervised Siamese network (SSiam). In TSiam, we utilize the tracking supervision to obtain the pair, additional we include negative training pairs for singleton tracks – tracks that are not

temporally co-occurring. As supervision from tracking may not always be available, to enable the use of metric learning without any supervision we propose an effective approach SSiam that can generate the required pairs automatically during training. In SSiam, we leverage dynamic generation of positive and negative pairs based on sorting distances (*i.e.* ranking) on a subset of frames and do not have to only rely on video/track based supervision.

Next, we present a method namely Clustering-based Contrastive Learning (CCL), a new clustering-based representation learning approach that utilizes automatically discovered partitions obtained from a clustering algorithm (FINCH) as weak supervision along with inherent video constraints to learn discriminative face features. As annotating datasets is costly and difficult, using label-free and weak supervision obtained from a clustering algorithm as a proxy learning task is promising. Through our analysis, we show that creating positive and negative training pairs using clustering predictions help to improve the performance for video face clustering.

We then propose a method face grouping on graphs (FGG), a method for unsupervised fine-tuning of deep face feature representations. We utilize a graph structure with positive and negative edges over a set of face-tracks based on their temporal structure of the video data and similarity-based constraints. Using graph neural networks, the features communicate over the edges allowing each track's feature to exchange information with its neighbors, and thus push each representation in a direction in feature space that groups all representations of the same person together and separates representations of a different person.

Having developed these methods to generate weak-labels for face representation learning, next we propose to learn compact yet effective representation for describing face tracks in videos into compact descriptors, that can complement previous methods towards learning a more powerful face representation. Specifically, we propose Temporal Compact Bilinear Pooling (TCBP) to encode the temporal segments in videos into a compact descriptor. TCBP possesses the ability to capture interactions between each element of the feature representation with one-another over a long-range temporal context. We integrated our previous methods TSiam, SSiam and CCL with TCBP and demonstrated that TCBP has excellent capabilities in learning a strong face representation. We further show TCBP has exceptional transfer abilities to applications such as multimodal video clip representation that jointly encodes images, audio, video and text, and video classification.

All of these contributions are demonstrated on benchmark video clustering datasets: *The Big Bang Theory*, *Buffy the Vampire Slayer* and *Harry Potter 1*. We provide extensive evaluations on these datasets achieving a significant boost in performance over the base features, and in comparison to the state-of-the-art results.

Acknowledgements

Firstly, I would like to thank my supervisor Rainer Stiefelhagen for his guidance, support, advice and most importantly, encouragement. I also thank Tamim Asfour for kindly agreeing to be a reviewer and helping to improve the quality of this thesis.

I thank my main collaborators Ali Diba, M. Saquib Sarfraz and Makarand Tapaswi. They have been an inspiration to me. Their daily and sometimes weekly conversations have proved to be one of my best learning experiences during my Ph.D. thesis.

Special thanks to Corinna Haas-Hecker for her help with the university's administrative things and making me feel welcome.

I thank my previous supervisor, Luc Van Gool at KU Leuven/ETH Zürich. He is one of the most intelligent and inspiring people I have known. He taught me how to think outside of the box and introduced me to the field of computer vision.

I would also like to thank my other advisors at Massachusetts Institute of Technology, Harvard Medical School and Harvard University. Ramesh Raskar is to be thanked for his brilliant ideas and inspiration. Rajiv Gupta inspired me to work harder and better on problems that can impact billions of lives. Mauricio Santillana inspired me to work on tracking epidemic outbreaks. I thoroughly enjoyed our weekly conversations to bounce ideas off of and debate with.

I thank all other collaborators including Davy Neven, Michael S. Brown, Gabriela Csurka, Naila Murray, Diane Larlus, Ali Pazandeh, Hamed Pirsiavash, Veith Röthlingshöfer, Mohsen Fayyaz, Manohar Paluri, Juergen Gall, Praneeth Vepakomma, Tristan Swedish, Ken Chang, Jayashree Kalpathy-Cramer, Congcong Wang, Faouzi Alaya Cheikh, Azeddine Beghdadi, Ole Jacob Elle, Jon Yngve Hardeberg, Sony

George, ... for your support and help. The final three dots should be interpreted as the people I forgot to mention here and as an apology.

Further, during my Ph.D. thesis, I was lucky and fortunate enough to find wonderful numerous friends. Without their company my social life would have been incomplete. I would also like to thank them. These include Keni Bernardin (the most available, reliable and lively party guy I have known. Without you, I would have never known the real music: James Brown, funky jazz, acid jazz, blues and many more, thanks for that!), Ertunc Ertekin (my unstoppable boxing coach), André Nordbø (my always reliable soundboard for discussions), Haroon Mughal, Constantin Seibold (after 4am we start working), Sebastian Bullinger, Piyush Verma, Abhishek Singh, Subhash Chandra Sadhu, Himi Mathur, Connor Henley (Boston nightlife belongs to you!), Samuel Schöb (The Muddy Charles Pub misses you!), Kevin Marty (my MIT social backbone), Fatima Villa (coffee tastes better when you're around), Guy Satat, Tjada Schult, Hao Li (the after MIT tech review party stays memorable!), Backtosch Mustafa (you indeed have impressive social skills - Harvard is complementary!), César Roberto de Souza, Uta Büchler, Patrick Buehler, Biagio Brattoli, Alessandra Bernardi, Rafael Rezende, Lagnojita Sinha, Tomohiro Maeda, Arun Nair (we share the pride of ACMMM'19 best paper together!), Eric Marengaux (you own Mont Aiguille!), Christoph Feichtenhofer, David Ross, Du Tran, Sourish Chaudhuri, Georg Buchner, Anca Nicoleta Ciubotaru, Pablo Galindo Zapata, Bert De Brabandere, Daniel Koester, Elsa Itambo, Amey Chaware, Ankit Ranjan, Syed Qasim Bukhari (the guy who knows the A-Z of human connections and communication), Ewa Szyszka, Prithvi Rajasekaran, Andres Mafla Delgado, Martin Humenberger (I agree, you told me, graduating earlier is better), Ferran Hueto Puig, Leonie Malzacher, Eva Schlindwein (my first awesome unofficial Ph.D. examiner - secret stays between us!), ...

Most importantly, I would like to thank my family for their inestimable support, understanding, love, care and believing in me.

Contents

List of Abbreviations

TSiam Track-supervised Siamese network . . . 5
SSiam Self-supervised Siamese network . . . 5
CCL Clustering-based Contrastive Learning . . . 6
FGG Face Grouping on Graphs . . . 6
TCBP Temporal Compact Bilinear Pooling . . . 6
MLP Multilayer Perceptron . . . 10
ReLU Rectifying Linear Unit . . . 10
CNN Convolutional Neural Network . . . 16
ELU Exponential Linear Unit . . . 16
SELU Scaled Exponential Linear Unit . . . 16
BBT The Big Bang Theory . . . 21
BF Buffy - The Vampire Slayer . . . 21
ACCIO Harry Potter 1 . . . 22
WCP Weighted Clustering Purity . . . 24
ACC Clustering Accuracy . . . 24
VAE Variational Autoencoder . . . 30
HAC Hierarchical Agglomerative Clustering . . . 35
BN Batch Normalization . . . 65
GCN Graph Convolutional Network . . . 78
FC Fully-Connected . . . 82
resGC residual Graph Convolution . . . 82
PCA Principal Component Analysis . . . 93

Basic Notations

Unless indicated otherwise, lower-case letters type-set in bold denote vectors, upper-case letters represent a matrix or tensor and lower-case letters represent scalars.

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Objective and motivation

In this thesis we propose several methods to fine-tune deep face features in a self-supervised manner. Specifically, we address the problem of discriminative face representation learning in TV series and movies for face recognition tasks. Face representation learning has attracted quite some attention, due to the potential applications in video understanding, video summarization, content-based indexing & retrieval, video descriptions for visually impaired people, and more. We believe the performance of discriminative learning is driven by two factors: the (base) feature representation and the learning algorithm. It has been shown, a good initialized feature representation is complementary to the learning algorithm. An ideal representation has small distances between samples from the same class, and large inter-class distances in feature space. While considerable progress has been made using the current state-of-the-art learning algorithm - deep neural networks, learning a good representation often requires a large-scale dataset with manually curated ground-truth labels. On top of the challenges that make face recognition hard, there are issues like facial expressions, pose variations, ageing, occlusion and more. To harness the power of deep networks on smaller datasets and tasks, pre-trained models (*e.g.* VGG-Face (Cao et al., 2018; Parkhi et al., 2015) trained on a large number of face images) are often used as a feature extractors or fine-tuned for the new task.

Deep neural networks to learn powerful features for face recognition can be categorized into two paradigms: a *fully-supervised* setting where ground-truth labels are often provided by humans (Cao et al., 2018; Parkhi et al., 2015; Schroff et al., 2015; Taigman et al., 2014), and a *self-supervised* learning paradigm where no ground-truth labels are used for model training rather weak-information that come with the data for free are used for learning a representation. (Bäuml et al., 2013; Zhang et al., 2016a,b). This thesis falls in the pool of *self-supervised* learning setting where we generate weak-labels or pseudo-labels by learning objectives properly so as to get supervision from the data itself to learn discriminative face representations.

In the *self-supervised* learning paradigm for learning discriminative face representations in videos, previously several attempts have been made. They mostly utilized pairwise constraints mined from face-tracks (Bäuml et al., 2013; Zhang et al., 2016a,b), such as the *must-link* constraint that two faces from the same track belong to the same person, and the *cannot-link* constraint that faces from tracks overlapping in time belong to different persons. These constraints are then used as a pairwise guide towards a suitable representation of each person in feature space using a loss function.

Fortunately, there is a vast amounts of video material on television and the Internet that can be used by machines to learn an effective representation that exhibits a small intra-person-distance, and large inter-person-distance in feature space. This thesis develops self-supervised methods that can learn a strong face representation from large-scale data be in the form of images or video. Specific contributions are described Section 1.3.

1.2 Key challenges and tasks

There are many key challenges and tasks in dealing with the application of face recognition, here we list a few.

1.2.1 Key challenges

Appearance variations. In video face clustering, grouping all faces of the same person into a unique cluster is challenging, due to of several reasons, such as (a) variations in facial expressions *e.g.* anger, disgust, fear, happiness, sadness or surprise;

(b) pose variations around egocentric rotation angles *i.e.* pitch, roll and yaw, or camera changing point of views' (c) variations in image resolution; (d) Presence or absence of structural components such as moustache, cap, and spectacles; (e) Partial face occlusion by foreground and background objects; (f) Ageing of the human face; and (g) Variations of illuminations, where low levels of lighting makes face detection and recognition hard, and high levels of lighting makes face overexposed. All the variations make the face clustering challenging.

Training data. For learning a good representation, training deep neural networks often requires a large-scale dataset with manually curated ground-truth labels. Annotation of video is expensive. For the approach to scale in the age of Netflix, YouTube and Instagram, where hundreds of hours of video material are uploaded to the internet *per minute*, it is vital that no manual annotation is required. The approach should furthermore not expect the ground-truth labels and rather utilize weak-information that come with the data for free for learning a good representation. In this direction, self-supervised learning has shown to be promising and usable in a general setting as collecting large-scale datasets is extremely expensive.

Speed and scalability. Nowadays, we have digital archives with huge collection of datasets, searching and indexing to find a certain specific person, object, or place is a common application. While retrieving specific instances through the video archival of hours of data and millions of frames, we expect the results to be real-time. Also, feature description is of paramount importance and needs careful design for representing faces in billions of images, such that they have: (a) Low computational overhead; (b) Extremely fast to compute; and (c) Scale to very large data and provide discrimination.

1.2.2 Key Tasks

Actor identification. An interesting multimedia application is actor identification, where the goal is to identify on-screen characters. While watching a movie or TV series, not always we remember all characters names in movie video. The objective is to identify on-screen character faces and label them with the corresponding names in the cast list. Usually, this automatic naming assignment is employed without any manual supervision and rather making use of textual cues, like cast lists, transcripts,

subtitles and closed captions. Netflix already provide these features and services to their customers.

Instances of person search. An important need in many conditions regarding video collections (archive video search/reuse, personal video organization/search, surveillance, law enforcement, protection of brand/logo use) is to locate more video segments of a certain precise person, object, or place, given a visual example. This is particularly useful on retrieving specific instances of persons in specific locations given a query image. This can save hours of efforts to manually look through the video archival.

Digital photo management. With the explosively increasing amount of personal photos, due to the rapid popularization of digital cameras and smart phones - there is a huge demand of digital photo management system for organizing photo collection. Recently several companies like Facebook, Google, Apple provide API services that provides a face-based album clustering and person search. The functionality of these API is to accurately and efficiently cluster photograph collections based totally on person identities, and group all faces of the same person into a small number of clusters.

Another rather unexpected application of digital photo management is in portrait collections, where the goal is to enable searching for portraits of famous identities. Portrait collection is of great importance to our cultural heritage as the collection of portraits tell extraordinary stories of encounter, exploration, independence, individuality and achievement of the famous identity.

1.3 Overview and Contributions

This thesis focuses on self-supervised face representation learning, wherein we propose methods to automatically generate weak-labels for training a neural network. Specifically, we propose several self-supervised approaches to generate positive and negative face pairs utilizing the temporal structure of the video and similarity-based constraints to learn discriminative face representations. In this section, we list the main contributions made in this thesis. First in Chapter 2, we give a brief introduction of neural networks training. Following in Chapter 3 we propose a method to mine pseudo-labels by sorting distances on a subset of face images. In Chapter 4, we

propose a way to generate weak labels from a clustering algorithm and show how they can be used efficiently to improve video face clustering. In Chapter 5 we look at graph neural networks for self-supervised face grouping on graphs, wherein we utilize the video constraints: must-link/cannot-link constraints to generate weak-labels for model training. Finally in Chapter 6 we propose Temporal Compact Bilinear Pooling (TCBP) to encode face tracks in videos into a single descriptor, which we then integrate with methods presented in Chapter 3 and Chapter 4 for learning a more powerful representation. The research covered in this thesis has resulted in several peer-reviewed publications. The contributions of each of these chapters are briefly discussed next.

Chapter 2 briefly overviews background on the neural networks from basics building blocks to hyper-parameter optimization via training using back-propagation algorithm. We also discuss briefly the most popular convolutional neural networks for image classification and face recognition tasks. The aim of this chapter is to provide the reader an overview how a neural network is trained, and knowledge that is frequently needed in this thesis.

Chapter 3 proposes two discriminative approaches to refine the face descriptors automatically: Track-supervised Siamese network (TSiam) and Self-supervised Siamese network (SSiam). In Track-supervised Siamese Network (TSiam), we utilize video-level constraints to generate a set of similar and dissimilar face pairs, and we further additionally include negative training pairs for singleton (non co-occurring) tracks by exploiting track-level distances. In Self-supervised Siamese Network (SSiam), we obtain hard positive and negative pairs by sorting distances on a subset of frames and not requiring video/track level constraints.

The content of this chapter is based on the following three publications:

- Vivek Sharma, M Saquib Sarfraz, and Rainer Stiefelhagen. **"A simple and effective technique for face clustering in tv series"**. In IEEE Computer Vision and Pattern Recognition (CVPR): Workshop on Brave New Motion Representations, 2017.

- Vivek Sharma, Makarand Tapaswi, M Saquib Sarfraz, and Rainer Stiefelhagen. **"Self-supervised learning of face representations for video face clustering"**. In IEEE International Conference on Automatic Face and Gesture Recognition (FG), 2019. *Oral presentation, Best paper award.*

- Vivek Sharma, Makarand Tapaswi, M Saquib Sarfraz, and Rainer Stiefelhagen. **"Video face clustering with self-supervised representation learning"**. IEEE Transactions on Biometrics, Behavior, and Identity Science (TBIOM), 2019.

In **Chapter 4** we propose Clustering-based Contrastive Learning (CCL), a new clustering-based representation learning approach that uses labels obtained from clustering along with inherent video constraints to learn discriminative face features. Specifically, we show that we can train discriminative models using positive and negative pairs obtained through clustering and video-level constraints that do not rely on face tracking.

The content of this chapter is based on the following publication:

- Vivek Sharma, Makarand Tapaswi, M Saquib Sarfraz, and Rainer Stiefelhagen. **"Clustering based contrastive learning for improving face representations"**. In IEEE International Conference on Automatic Face and Gesture Recognition (FG), 2020.

In **Chapter 5** we look at graph neural networks to learn face representations. We propose Face Grouping on Graphs (FGG), an approach to self-supervised face grouping, where edges represent one of must-link/cannot-link constraints that emulates the communication. Using a neural network that operates on the graph structure and differentiates between the different edge types, the features can exchange information and assimilate their position in feature space compared to other features belonging to the same person. Additionally, we propose to work with partially pooled features as a trade-off between robustness on track-level and conservation of variance information on frame-level.

The content of this chapter is based on the following publication:

- Veith Röthlingshöfer*, Vivek Sharma*, and Rainer Stiefelhagen. **"Self supervised face-grouping on graphs"**. In ACM International Conference on Multimedia, 2019. *Spotlight: oral presentation.*

In **Chapter 6** we propose Temporal Compact Bilinear Pooling (TCBP) an extension of the Tensor Sketch projection algorithm (Pham and Pagh, 2013) to incorporate a temporal dimension for representing face tracks in videos. We show that encoding

face tracks into a single descriptor are powerful in representing faces in images and videos.

The overall theme of this chapter is to learn multimodal clip representation that jointly encodes images, audio, video, and text using TCBP for video ordering task. Additionally, we show that TCBP features show exceptional transfer abilities to applications such as video classification, video retrieval and face video face clustering.

The content of this chapter is based on the following publication:

- Vivek Sharma, Makarand Tapaswi, and Rainer Stiefelhagen. **"Deep multimodal feature encoding for video ordering"**. In IEEE International Conference on Computer Vision (ICCV): workshop on Large Scale Holistic Video Understanding, 2019. *Oral presentation.*

Finally, **Chapter 7** concludes the thesis with a summary of the contributions, limitations and directions for future work.

Chapter 2

Background

In this chapter, we briefly overview the background information and theory related to the works presented in the manuscript. We first introduce to neural networks. We then explain the parameter estimators and learning: loss function, regularizer and training of neural networks. Finally, we present the popular neural network architectures, datasets and evaluation metrics used frequently in this thesis.

2.1 Neural Networks

Artificial neural network or neural network is the heart of the deep learning techniques. Neural networks are loosely inspired by neuroscience - biological neurons, and are designed to recognize patterns. The patterns they recognize are numerical, typically vector-valued. The neural networks can recognize patterns from the real-world data be it in the form of images, sound, text or time-series.

Given a training dataset with N samples $\mathcal{D} = \{\mathbf{x}_i, \boldsymbol{c}_i\}_{i=1}^{N}$, where $\mathbf{x} \in \mathbb{R}^m$, and $\boldsymbol{c}_i$ is the vector of target labels corresponding to the input sample $\mathbf{x}_i$. A neural network is employed to learn a function f that maps an input $\mathbf{x}_i$ to target-output $\boldsymbol{c}_i$. They are called networks because the learned function can be composed of multiple functions, connected in chain to form *e.g.* $f(\mathbf{x}) = f_3(f_2(f_1(\mathbf{x})))$, where f_1 is the first layer of the network, f_2 is the second layer and f_3 is the output layer of the network. c_i represents the desired output of the f_3 layer, while f_1 and f_2 are hidden layers and does not represent the desired output. Each layer f_k is parameterized by a learnable weight vector $\mathbf{w}_k$, $f_k(\mathbf{x}) = \phi(\mathbf{w}_k\mathbf{x})$, where ϕ is the activation function. The layer can be

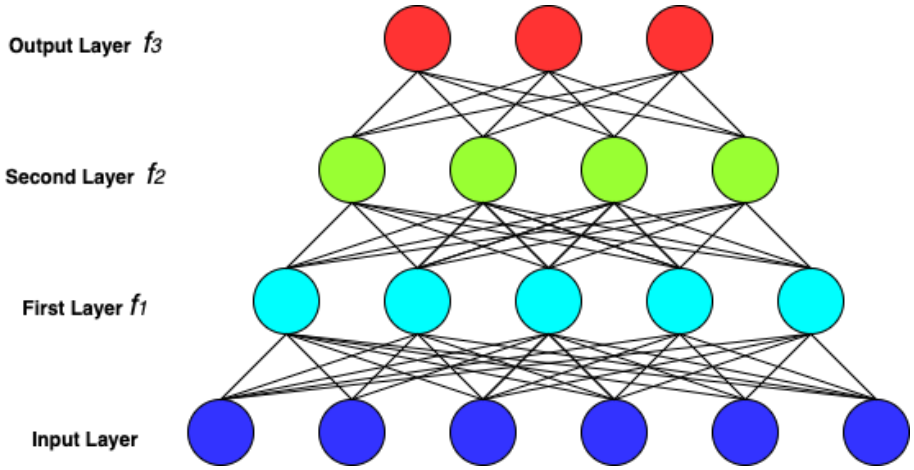


Figure 2.1: Multilayer perceptron (MLP): an example neural network scheme with 3 layers.

viewed as a group of neurons, also called perceptrons where each neuron has inputs and learnable weights for each input that maps the vector input to a scalar output, also referred as Multilayer Perceptron (MLP). The activation function ϕ serves as a non-linearity, currently used most popular activation function being Rectifying Linear Unit (ReLU) (Glorot et al., 2011). They are used depending on the desired range of the neuron's output, although other functions are also investigated. Neural networks can be seen as a universal approximator, meaning that if provided with enough neurons in the hidden layer and a suitable activation function. The learning of the neural network occurs through minimizing a loss function, i.e. the difference between the predicted output and the target output of the neural network. Because of the non-linearity of a neural network causes loss functions to become non-convex, neural networks are optimized by using iterative gradient descent steps on batches randomly sampled from the training set. The gradients of the loss function are computed using the back-propagation algorithm (Rumelhart et al., 1985) and the chain rule, meaning that we start from the last layers and backpropagate the estimated loss towards the preceding layers of the neural network. All the samples are shown during the training of the neural network in each epoch, and the process is repeated until convergence is reached. After a local minima is attained, the trained neural network is deployed for testing new samples that were not shown during the training phase. For a more extensive review on neural networks, we refer the reader to Goodfellow et al. (2016). Figure 2.1 illustrates an example neural network scheme with 3 layers.

Often, matrix notations is used to describe the operations in a neural network. The weight of the layer f_k is given as a weight matrix $W \in \mathbb{R}^{n \times m}$ that transforms the

input $\mathbf{x} \in \mathbb{R}^n$ to $\mathbf{y} \in \mathbb{R}^m$ following through the activation function ϕ. An additional parameter, bias term ($\mathbf{b}$) in the neural network is also added which is used to adjust the output along with the weighted sum of the inputs to the neuron, $\mathbf{b} \in \mathbb{R}^m$.

$$\mathbf{y} = \phi(W\mathbf{x} + \mathbf{b}) \tag{2.1}$$

During training a neural network, we aim to learn a set of weights for the neurons that minimizes the divergence between the predicted and target output using an appropriate loss function. The loss functions are task specific and are user defined.

2.2 Learning: Parameter Estimators

Although in this thesis, we focus on self-supervision only. Here, we explain learning for supervised setups aswell.

2.2.1 Loss Function

Regression loss. The aim of this loss is to predict a continuous value. Euclidean loss or L2-loss is the most commonly used regression loss function. L2 loss is the squared distance between the target output $\mathbf{y}$ and the network prediction $\hat{\mathbf{y}}$. The L2 loss is given as:

$$\mathcal{L} = \|\hat{\mathbf{y}} - \mathbf{y}\|_2^2 \tag{2.2}$$

L1 loss computes absolute differences between the target output and the network prediction. The loss is defined as:

$$\mathcal{L} = \|\hat{\mathbf{y}} - \mathbf{y}\|_1 \tag{2.3}$$

Other popular regression loss functions *e.g.* Huber loss (Huber, 1992).

Classification loss. The aim of this loss function is to predict categorical value. The commonly used loss for classification is cross-entropy loss.

The softmax function is given as:

$$P_q = \frac{\exp(a_q)}{\sum_{r=1}^{C} \exp(a_r)} \tag{2.4}$$

The cross entropy loss also known as softmax loss is defined as:

$$\mathcal{L} = -\sum_{q=1}^{C} y_q \log(P_q) \tag{2.5}$$

where a is the output of the last layer of neural network that is fed to a C-way softmax function, $\mathbf{y}$ is the vector of target labels for input data $\mathbf{x}$, and C is the number of output classes.

Metric loss. The objective of this metric loss function is not to classify input samples, but to learn distances that exhibits small distances between samples for similar observations, and large distances for different ones. The commonly used metric loss functions are contrastive loss and triplet loss.

Contrastive loss. The contrastive loss (Hadsell et al., 2006) encourages small distances between samples from the same label, and far apart at least by the margin for the samples of different labels in feature space. The contrastive loss is given as follows:

$$\mathcal{L} = \frac{1}{2}\left((1-y)\cdot(d_W)^2 + y\cdot(\max(0, m-d_W))^2\right) \tag{2.6}$$

where d_W is the euclidean distance between the sample pair $\mathbf{x}_1$, $\mathbf{x}_2$ with $y = 0$ for a positive pair and $y = 1$ when corresponding to a negative pair, and m is the margin.

Triplet loss. The triplet loss (Schroff et al., 2015) encourages the distances between anchor and positive pair to be less than the distance between the anchor and negative by atleast distance margin. Triplet loss comprises of a pull-term, pulling data points from the same class closer, and a push-term, pushing data points from a different class further away. The triplet loss is formulated as:

$$\mathcal{L} = max(d(\mathbf{x}_1, \mathbf{x}_2) - d(\mathbf{x}_1, \mathbf{x}_3) + m, 0) \tag{2.7}$$

where d is euclidean or cosine distance on the embedding space, m is the margin, and $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$ are anchor, positive, negative respectively. $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$ are anchor, positive of the same class as the anchor, negative of a different class respectively.

Other losses. There are several custom designed loss functions, specialized for different tasks, such as for geometric problems (Kendall and Cipolla, 2017), instance segmentation (Berman et al., 2018; Briggman et al.; De Brabandere et al.), discriminative face representation learning (Chopra et al., 2005; Schroff et al., 2015; Zhang et al., 2016a), image-caption retrieval (Vendrov et al., 2016) and many more tasks. In Chapter 3 and Chapter 4, we propose loss functions for discriminative face representation learning and in Chapter 6 we propose to utilize a loss function for video representation learning.

2.2.2 Regularizer

A model that learns the train set too well and performs very poorly on the test set is a sign of overfitting. In such a case, the neural network has a very high variance and it cannot generalize well to the test data. Common ways to reduce overfitting are, (a) getting more training data; (b) use regularization to control the model variance. The commonly used regularization methods for neural networks are L2 regularization and dropout.

L2 regularization is also known as ridge regression or Tikhonov regularization. In L2 regularization, ℓ_2 norm penalty is added to the cost function that penalizes large weights, and thus aim at limiting the model capacity.

Dropout randomly ignores nodes from a neural network during training to regularize it. The nodes are "dropped-out" randomly. That means other available neurons handle the representation required to make predictions. Thus, for each iteration, unique internal representations are learned by the network. In this way, the network becomes less sensitive to the specific weights of neurons, which in turn leads to better generalization and also helps to overcome from over-fitting.

2.2.3 Optimization

Once a neural network model, a loss function and a regularization is chosen, we can optimize the neural network to find the network parameters θ^* that minimizes the cost function on the entire training set:

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \sum_i \mathcal{L}_i(f_\theta(\mathbf{x}_i), \mathbf{y}_i) \tag{2.8}$$

The network parameters $\theta = (\mathbf{w},\mathbf{b})$, where $(\mathbf{w},\mathbf{b})$ is the network weight and bias. The loss function computes loss for a single training example, while the cost function is average loss for the entire training set.

The most simple and naive way to optimize the neural network is to compute the derivative of the loss function wrt. the entire training set and then backpropagate the gradient through the preceding layers and update the network parameters. It is naive because of two reasons,

(a) It only relies on the first order information (*i.e.* the gradient). This can be addressed using different normalisation techniques.

(b) For computing the gradient it relies on the whole training set which is very expensive and practically not possible to compute for a large-scale dataset. This is usually addressed using a stochastic gradient descent (SGD) approximation.

Gradient Descent. The optimization is done using an iterative gradient descent optimization algorithm that makes use of the gradients to find local minima of the loss function *i.e.* iteratively moving in the direction of steepest descent. At each iteration, the gradients on each weight are computed using the back-propagation algorithm. Then, we update the parameters of the network, by taking a small step in the direction as defined by the negative of the gradient:

$$\theta^{(t+1)} = \theta^{(t)} - \eta \frac{\partial \mathcal{L}}{\partial \theta^{(t)}} \tag{2.9}$$

where η is the learning-rate and it determines the size of the step. η should be chosen carefully: if a very low learning rate is used, this may lead to slow convergence, and if a high learning rate is used, this may risk overshooting the lowest point and thus the training may not converge at all. SGD was used extensively in the thesis.

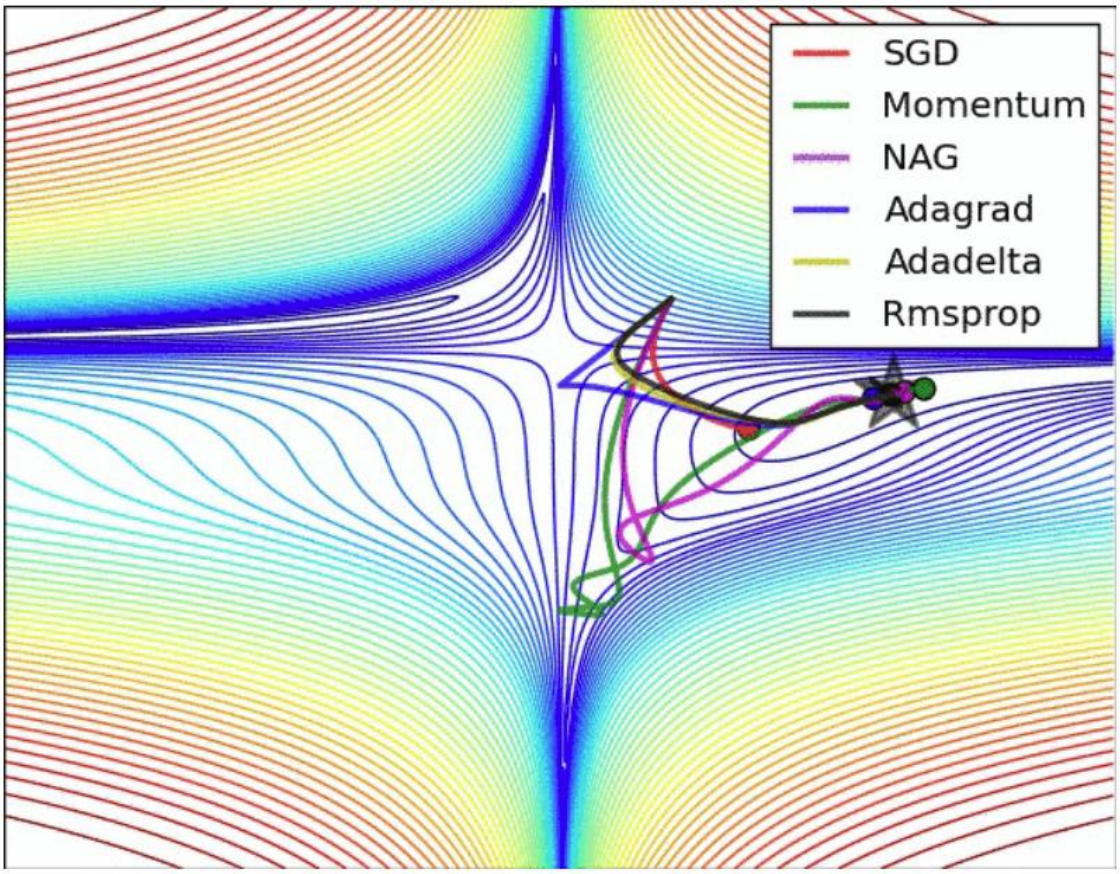


Figure 2.2: Landscape of popular optimizers. Figure taken from Ruder (2016)

Other popular optimizers are Adagrad, Adam, Adadelta, RMSProp (Duchi et al., 2011; Kingma and Ba, 2015; Qian, 1999; Zeiler, 2012). For a more detailed review on optimizer, we refer the reader to Ruder (2016). Figure 2.2 shows the convergence behaviour of popular optimizers.

Back-Propagation. The back-propagation algorithm (Rumelhart et al., 1985) also known as backprop, allows to compute the gradients. Back-propagation is the method for computing the gradients of the loss function, while SGD is used to perform learning this gradient. Back-propagation adjusts each weight in the neural network in proportion to the loss function. Back-propagation is merely an application of the chain rule to find the derivatives of loss function with respect to the network parameters. The process of computing the gradients passing through preceding n

layers with activations y_i upto i-th layer for each weight w_i from the loss function $\mathcal{L}$ is known as back-propagation, given as:

$$\frac{\partial \mathcal{L}}{\partial w_i} = \frac{\partial \mathcal{L}}{\partial y_n} \cdot \frac{\partial y_n}{\partial y_{n-1}} \cdots \frac{\partial y_{i-2}}{\partial y_{i-1}} \cdot \frac{\partial y_{i-1}}{\partial y_i} \cdot \frac{\partial y_i}{\partial w_i} \tag{2.10}$$

In the next section, we discuss the convolutional neural networks, and the commonly used neural network architectures for image classification and face recognition.

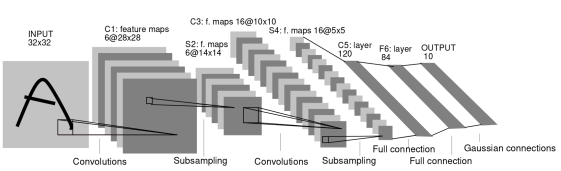
2.2.4 Convolutional Neural Network (CNN)

Convolutional neural networks (LeCun et al., 1989) are simply an extension of multilayer perceptrons. Convolutional neural networks are also known as ConvNets. The basic difference between MLPs to ConvNets is that the linear layers are replaced by convolutional layers. ConvNets are easier to train and have shown to generalize better than feedforward networks or MLPs for vision tasks.

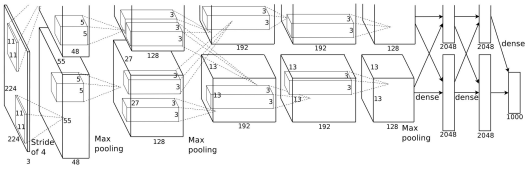
A Convolutional Neural Network (CNN) is composed of a stack of convolutional filter banks, each followed by the activation layer or non-linearity layer, and the pooling function or feature pooling layer. Each convolution layer takes an input feature map and outputs a new feature map through a set of weights called filter banks. In the convolution layer, the weights are shared that reduces the number of trainable parameters, and also since the feature map share the filter bank this benefits the convolution layers to be translation equivariant, meaning if the input changes, the output changes in the same way. Following the convolution layer, similar to MLPs we use non-linearity layer, most popular being ReLU (Glorot et al., 2011), other popular activations functions are Sigmoid, Tanh, Exponential Linear Unit (ELU), Scaled Exponential Linear Unit (SELU) and Swish (Clevert et al., 2016; He et al., 2015; Klambauer et al., 2017; Maas et al., 2013; Ramachandran et al., 2017)

Furthermore, pooling layers are used to introduce invariances in the network. Further, the other important benefit of pooling layers, is that they usually also downsample the feature maps to reduce dimensionality. Pooling layers are used together with convolution layers, the commonly use pooling functions are max pooling and average pooling.

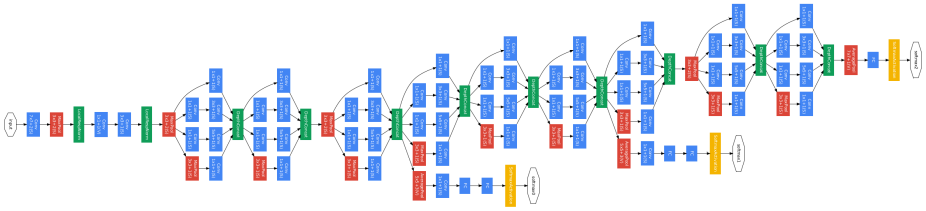
The output of a ConvNet is usually fed as an input to MLPs or a linear classifier.



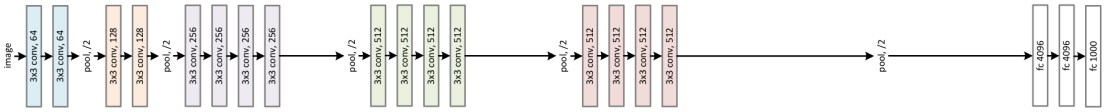
(a) LeNet-5 (LeCun et al., 1989)



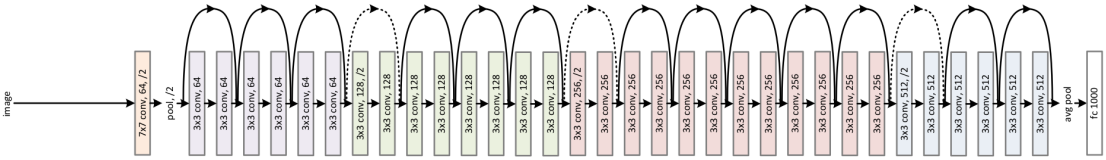
(b) AlexNet (Krizhevsky et al., 2012)



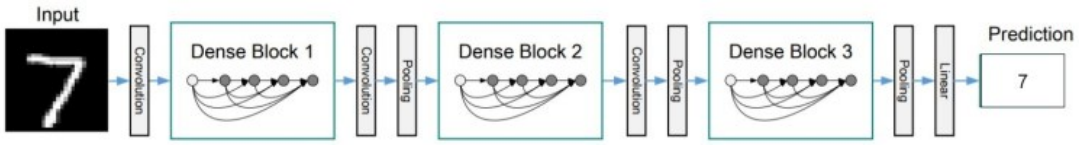
(c) GoogLeNet (Szegedy et al., 2015)



(d) VGG-19 (Simonyan and Zisserman, 2014b)



(e) ResNet-34 (He et al., 2016)



(f) DenseNet (Huang et al., 2017)

Figure 2.3: Common neural network architectures.

Now we briefly overview the commonly used neural network architectures for image classification and face recognition tasks.

Common Neural Network Architectures. Image classification is commonly used for investigating efficiency/quality trade-offs of ConvNets in computer vision community. Below we discuss the most popular networks.

LeNet. LeNet (LeCun et al., 1989) was the first convolutional neural network for classification of hand-written digits. LeNet is a 7-level convolutional network, it

consists of two convolutional layers with sigmoid non-linearity activation functions, two pooling layers and a fully connected layer, see Figure 2.3a.

AlexNet. AlexNet (Krizhevsky et al., 2012) won the ILSVRC 2010 competition on the large-scale ImageNet dataset (Russakovsky et al., 2015) by reducing the top-5 error from 26% to 15.3%. AlexNet is composed of 5 convolutional layers followed by 3 fully connected layers, see Figure 2.3b. AlexNet is very similar to LeNet but was stacked with more layers to obtain a deeper network and also with more filters per layer. They replaced the sigmoid with ReLU activations.

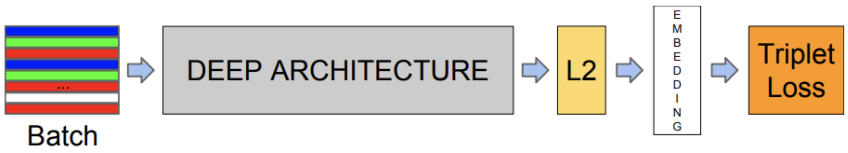
GoogLeNet. GoogLeNet (Szegedy et al., 2015) won the ILSVRC 2014 competition by reducing the top-5 error to 6.67%. In GoogLeNet, the authors proposed the inception module. The inception module performs convolution on an input with several very small convolutions (1x1, 3x3, 5x5) in order to drastically reduce the number of parameters. GoogLeNet is composed of 22 layers, see Figure 2.3c. Further, in GoogLeNet the authors propose for the first time a structured approach of stacking uniform modules.

VGG. VGG (Simonyan and Zisserman, 2014b) consists of 16 to 19 convolutional layers, and is similar to AlexNet. In difference to AlexNet, the author use only 3x3 convolutions uniformly throughout the architecture, see Figure 2.3d.

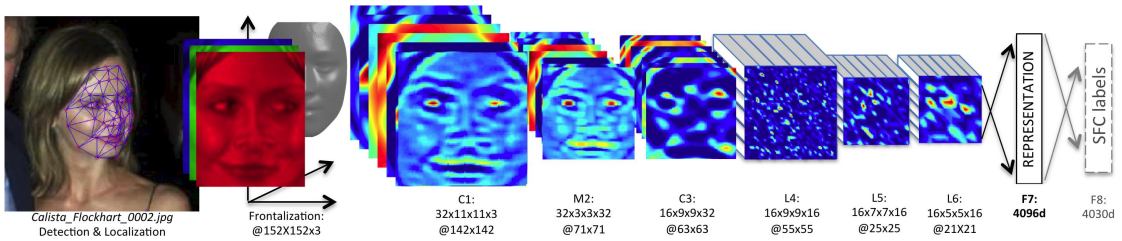
ResNet. ResNet (He et al., 2016) won the ILSVRC 2015 competition by achieving a top-5 error rate of 3.57% which beats human-level performance. In ResNet, the authors introduced a novel architecture with skip connections or residual connections. Skip connections are also known as gated recurrent units and share a lot of relevance from Recurrent Neural Networks (RNNs), see Figure 2.3e. ResNet has lower complexity than VGG architectures.

DenseNet. DenseNet (Huang et al., 2017) is a variant of ResNet. In DenseNet, there is a dense connectivity that directly connects the output of any layer to all subsequent layers within a module. This encourages feature re-use and and makes the network simpler and highly parameter efficient, see Figure 2.3f.

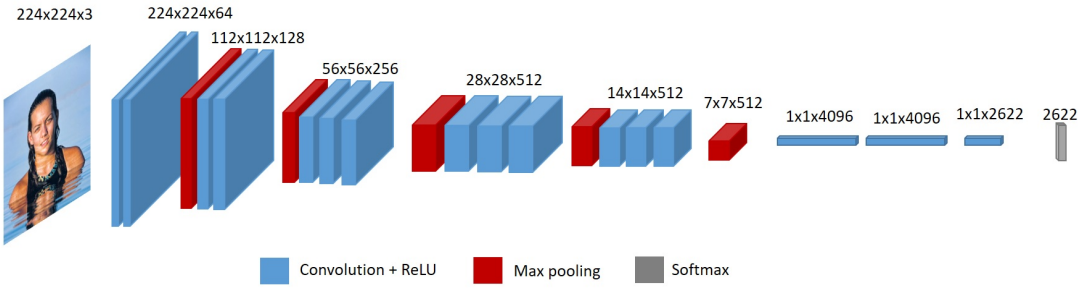
Face Recognition Networks. Another important application, where ConvNets have been very well deployed is for face recognition, where the general task is to identify faces present in images and videos.



(a) FaceNet (Schroff et al., 2015)



(b) DeepFace (Taigman et al., 2014)



(c) VGGFace (Parkhi et al., 2015). Image source: Google search.

Figure 2.4: Common face recognition neural network architectures.

FaceNet. FaceNet (Schroff et al., 2015) is a convolutional neural network for tasks such as face recognition, verification and clustering. FaceNet directly learns a mapping from face images to a compact Euclidean space. To train the network, the authors use triplet loss with anchor, positive and negative examples, see Figure 2.4a.

DeepFace. DeepFace (Taigman et al., 2014) is also a deep neural network for the task of face recognition. It employs a nine-layer neural net. DeepFace is trained using a dataset of 4 million facial images belonging to around four thousand people. The model is trained using cross-entropy loss. The conventional pipeline of DeepFace consists of four stages: detect, align, represent, and classify to achieve the face recognition, see Figure 2.4b.

VGGFace. VGGFace (Parkhi et al., 2015) is based on the VGG CNN architecture and VGGFace2 (Cao et al., 2018) is based on the ResNet CNN architecture. Both face

recognition models are trained on a large scale face dataset: with 2.6 million images of ∼2,600 subjects (VGGFace), and 3.31 million images of 9,131 subjects (VGGFace2). To train the network, the authors use cross-entropy loss, see Figure 2.4c. The models are freely available to the research community, unlike FaceNet and DeepFace which are not available publicly. In this thesis, we use VGGFace and VGGFace2 as the base model to extract face features.

Next we discuss the datasets and evaluation metrics used in this thesis.

2.3 DATASETS

We provide a short description and notation of datasets repeatedly in this thesis. When referring to a specific episode, we stick to the scheme *"[dataset]-[season][episode]"*.

We conduct experiments on three challenging face clustering datasets, namely *The Big Bang Theory* (BBT) (Wu et al., 2013a; Zhang et al., 2016a), *Buffy - The Vampire Slayer* (BF) (Zhang et al., 2016b), and *Harry Potter 1* (ACCIO) (Ghaleb et al., 2015).

We follow the protocol used in several recent video face clustering works (Cinbis et al., 2011; Wu et al., 2013b; Zhang et al., 2016a,b) that focus on improving feature representations for video-face clustering. First, they assume the number of main characters/clusters is known. Second, as the labels are obtained automatically, we learn episode specific embeddings. We also use the same number of characters as previous methods (Zhang et al., 2016a,b), however, it is important to note that we do not discard tracks/faces that are small or have large pose variation. We use the face tracks released by Bäuml et al. (2013) that incorporate several detectors to encompass all pan angles and in-plane rotations up to 45 degrees. Tracks are created via an online tracking-by-detection scheme with a particle filter.

Below we present key information about the datasets. In particular, note that previous works use much smaller datasets. Additionally, it is important to note that different characters have wide variations in the number of tracks, indicated by the cluster skew between largest class (LC) to smallest class (SC).

Figure 2.5: Example images for a few characters from our dataset. We show one easy sample and one difficult sample. The extreme variation in illumination, pose, resolution, and attributes (spectacles) make the datasets challenging.

2.3.1 The Big Bang Theory

The first dataset we evaluate is The Big Bang Theory (BBT) (Bäuml et al., 2013; Zhang et al., 2016a), containing face-tracks from the first six episodes of the first season. Traditionally, publications evaluating on BBT use the very first episode for reporting their performance. Since most scenes in BBT are set indoors with studio-lighting, the face-tracks are generally cleaner than in other datasets. There are five main characters, namely Sheldon, Leonard, Penny, Raj and Howard. Kurt is a sixth named character who does not count as a main character. We use the main characters for evaluation unless stated otherwise and discard all face-tracks of unknown characters, face-tracks labeled as false-positives and face-tracks containing track-switches (i.e. the track jumps from one character to another without interruption). Other works (Cinbis et al., 2011; Wu et al., 2013b; Xiao et al., 2014; Zhang et al., 2016a) use fewer tracks; many profile tracks were removed. Table 2.1 shows statistics from BBT and exemplary faces can be found in Figure 2.5.

2.3.2 Buffy - The Vampire Slayer

Buffy - The Vampire Slayer (BF) is a TV series about vampires, and thus naturally contains many shots in the dark as well as action shots including motion blur or obstructed faces. The dataset (Bäuml et al., 2013; Zhang et al., 2016b) contains the first six episodes of the fifth season. The episode used for evaluation in other works is the second episode, i.e. BF-0502. There are a total of six main characters: Xander, Buffy, Dawn, Anya, Willow and Giles. Another twelve named characters exist. Similar to BBT, we use all tracks of the main characters and discard all other

Table 2.1: Statistics for each episode of The Big Bang Theory (Bäuml et al., 2013; Tapaswi et al., 2012; Wu et al., 2013a; Zhang et al., 2016a) (5 main characters), Buffy (Bäuml et al., 2013; Zhang et al., 2016b) (6 main characters) and ACCIO (Ghaleb et al., 2015) (35 main characters). #TR and #FR is the number of face-tracks and total faces. #Overlaps is the number of tracks that overlap in time. LC/SC indicates the largest and smallest ground truth cluster with non-main characters removed. (Cinbis et al., 2011; Wu et al., 2013a,b; Xiao et al., 2014; Zhang et al., 2016a) use fewer tracks (last column).

	This work			Previous work
Dataset	#TR (#FR)	LC/SC (%)	#Overlaps	#TR (#FR)
BBT-0101	644 (41220)	39.3 / 4.0	313	182 (11525)
BBT-0102	613 (32513)	35.1 / 8.6	302	—
BBT-0103	530 (30626)	44.0 / 7.7	145	—
BBT-0104	449 (27856)	42.1 / 8.5	160	—
BBT-0105	404 (26418)	34.4 / 5.7	154	—
BBT-0106	623 (40713)	31.3 / 13.6	443	—
BF-0501	552 (37758)	42.6 / 5.1	132	—
BF-0502	568 (39263)	34.0 / 5.9	173	229 (17337)
BF-0503	806 (40546)	30.9 / 1.9	211	—
BF-0504	288 (24429)	57.4 / 3.6	44	—
BF-0505	543 (34918)	53.7 / 4.1	91	—
BF-0506	535 (29340)	31.2 / 8.6	232	—
ACCIO	3243 (166885)	30.7 / 0.06	1799	3243 (166885)

tracks in the dataset. Again, previous works (Cinbis et al., 2011; Wu et al., 2013b; Xiao et al., 2014; Zhang et al., 2016a) have evaluated on a smaller subset of the dataset containing less face-tracks of the main characters. In Table 2.1 statistics from each episode of BF are shown. Figure 2.5 provides examples.

2.3.3 HARRY POTTER 1

The final dataset on which we evaluate is the first movie from the "*Harry Potter*" movie series (Ghaleb et al., 2015). Harry Potter 1 (ACCIO) contains a large number of dark scenes and several tracks with non-frontal faces. In the course of the movie the characters appear in many diverse lighting conditions, many of which are in dark, windowless rooms or lit by fire. In contrast to the other two datasets, ACCIO contains many more main characters, namely 35. Many of them never appear on

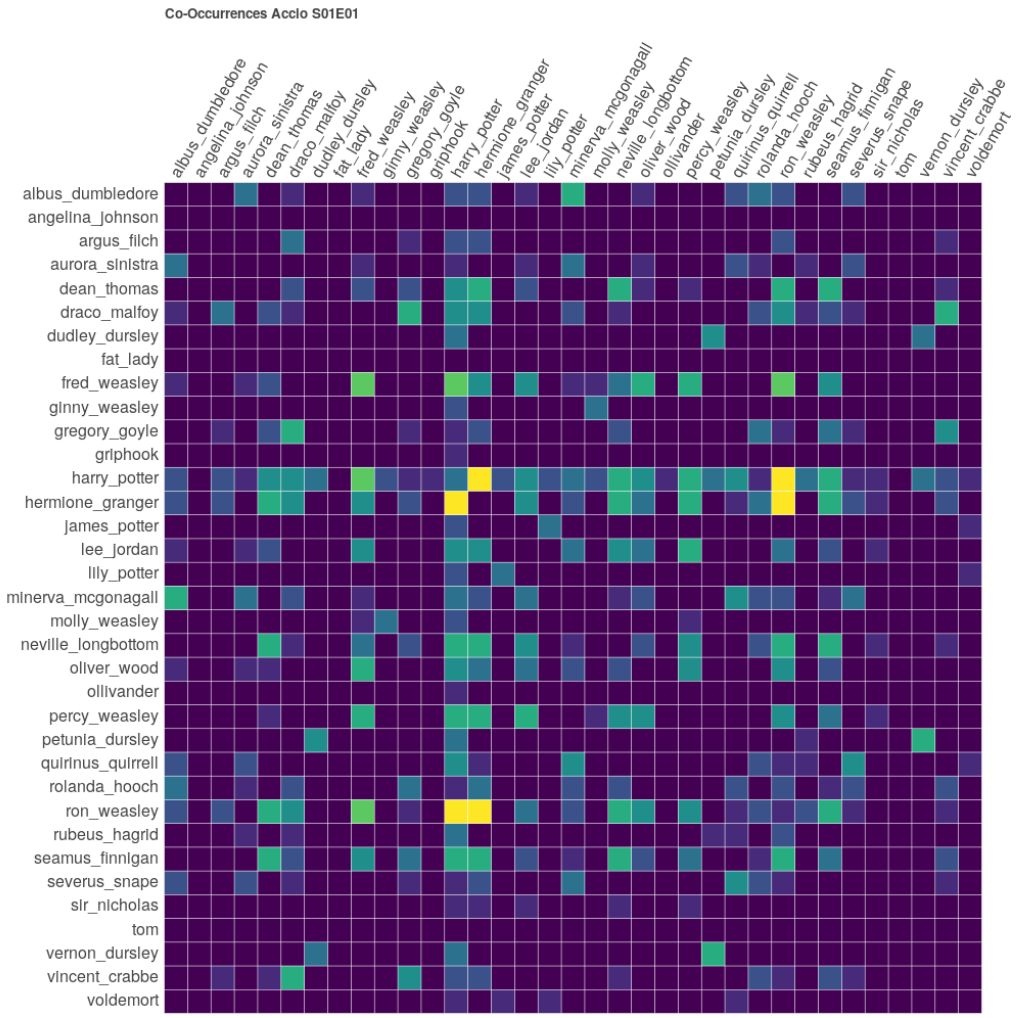


Figure 2.6: Co-occurrences resulting in temporal cannot-link edges on ACCIO. Brighter cells appear together more often. Table is log-scale. The regularly scaled table only has very few distinguishable cells between Harry, Ron and Hermione.

screen simultaneously as can be seen in the co-occurrence matrix in Figure 2.6. This implies that there is no cannot-link edge between these characters.

The difference in percentage of face-tracks belonging to the most frequently occurring character (Harry Potter) and the least frequently occurring character is two orders of magnitude greater compared to BBT and BF as shown in Table 2.1. Exemplary faces are shown in Figure 2.5.

Further, note that, in accordance with previous literature (Zhang et al., 2016b), for evaluation of ACCIO throughout the thesis, we use 36 and 40 clusters for the 35 main characters.

2.4 METRICS

Clustering Accuracy. The metric with which we evaluate our performance on BBT and BF is Weighted Clustering Purity (WCP) (Tapaswi et al., 2014b), also known as Clustering Accuracy (ACC) (Zhang et al., 2016b). It is computed by assigning the most common ground truth label within a cluster to all elements in that cluster: As we compare methods that generate equal numbers of clusters (number of main cast), ACC is a fair metric for comparison.

$$\text{ACC} = \frac{1}{N} \sum_{c=1}^{|C|} n_c \cdot p_c \,, \tag{2.11}$$

where N is the total number of tracks in the video, n_c is the number of samples in the cluster c, and cluster purity p_c is measured as the fraction of the largest number of samples from the same label to n_c. $|C|$ corresponds to the number of main cast members, and in our case also the number of clusters.

Additionally, we also report the clustering accuracy for ACCIO.

B^3 measures. The metric with which we evaluate our performance on ACCIO is BCubed (B^3) Precision (P), Recall (R) and F-measure (F) (Amigó et al., 2009; Bagga and Baldwin, 1998; Moreno and Dias, 2015). We used BCubed metrics in order to have a fair comparison with previous work Zhang et al. (2016b). BCubed metrics (Amigó et al., 2009) is the best suited technique for evaluating clustering performance with imbalanced class distribution. This is because B^3 metrics estimate precision and recall for each element of the cluster in comparison to the clustering purity, where the most common ground truth label is assigned to all elements within the cluster for computing purity. For each element e, we can define the correctness of the relation from that element to another element e' (Amigó et al., 2009):

$$correctness(e, e') = 1[L(e) = L(e') \Leftrightarrow C(e) = C(e')] \tag{2.12}$$

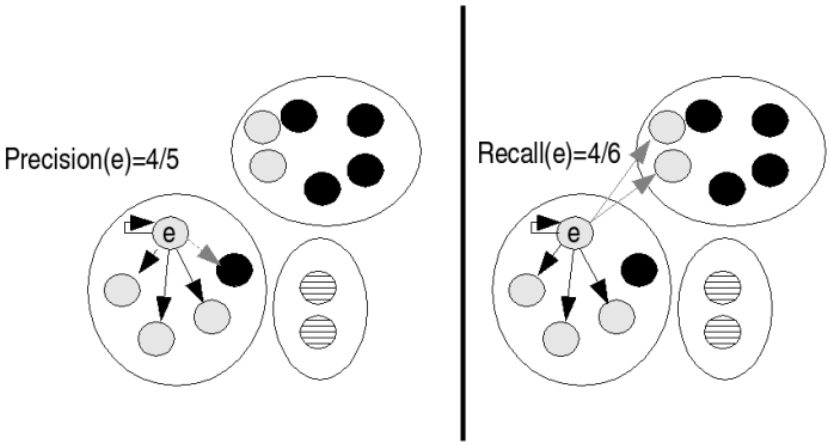


Figure 2.7: Computation of B^3 precision and recall (Amigó et al., 2009) for one element of the datasets. The value for the full dataset is the average over all elements.

Here 1 is the indicator function, L is the ground-truth label of the element, and C is the cluster to which the element is assigned. B^3 precision and recall over the full dataset are then defined as

$$B^3 Precision = \underset{e}{Avg}[\underset{C(e)=C(e')}{\underset{e'}{Avg}}[Correctness(e, e')]] \tag{2.13}$$

$$B^3 Recall = \underset{e}{Avg}[\underset{L(e)=L(e')}{\underset{e'}{Avg}}[Correctness(e, e')]] \tag{2.14}$$

The B^3 F-Score is the harmonic mean of these values. Figure 2.7 shows how to compute recall and precision for one element. The B^3 score has several advantages over WCP discussed in Amigó et al. (2009). We follow the example of previous work (Zhang et al., 2016b) and evaluate with 36 and 40 clusters on the ACCIO dataset.

For all experiments, unless stated otherwise, clustering evaluation is performed at track-level *i.e.* mean-pool of frames.

Chapter 3

Ranking-based Pair Generation

Characters are a key component of understanding the story conveyed in TV series and movies. With the rise of advanced deep face models, identifying face images may seem like a solved problem. However, as face detectors get better, clustering and identification need to be revisited to address the increasing diversity in facial appearance. In this chapter, we propose unsupervised methods for feature refinement with application to video face clustering. Our emphasis is on distilling the essential information, *identity*, from the representations obtained using deep pre-trained face networks. We propose a self-supervised Siamese network that can be trained without the need for video/track based supervision, that can also be applied to image collections. We evaluate our methods on three video face clustering datasets. Thorough experiments including generalization studies show that our methods outperform current state-of-the-art methods on all datasets. The datasets and code are available at *https://github.com/vivoutlaw/SSIAM*.

The content of this chapter is based on the following three publications:

- Vivek Sharma, M Saquib Sarfraz, and Rainer Stiefelhagen. **"A simple and effective technique for face clustering in tv series"**. In IEEE Computer Vision and Pattern Recognition (CVPR): Workshop on Brave New Motion Representations, 2017.

- Vivek Sharma, Makarand Tapaswi, M Saquib Sarfraz, and Rainer Stiefelhagen. **"Self-supervised learning of face representations for video face clus-**

tering". In IEEE International Conference on Automatic Face and Gesture Recognition (FG), 2019. *Oral presentation, Best paper award.*

- Vivek Sharma, Makarand Tapaswi, M Saquib Sarfraz, and Rainer Stiefelhagen. **"Video face clustering with self-supervised representation learning"**. IEEE Transactions on Biometrics, Behavior, and Identity Science (TBIOM), 2019.

3.1 INTRODUCTION

Long videos such as TV series episodes or movies are often pre-processed via shot and scene change detection to make the video more accessible. In recent years, person clustering and identification are gaining importance as several emerging research areas (Rohrbach et al., 2017b; Tapaswi et al., 2016; Vicol et al., 2018; Zhou et al., 2018) can benefit from it. For example, in video question-answering (Tapaswi et al., 2016), most questions center around the characters asking who they are, what they do, and even why they act in certain ways. The related task of video captioning (Rohrbach et al., 2017b) often uses a character agnostic way (replacing names by *someone*) making the captions very artificial and uninformative (*e.g.* *someone* opens the door). However, recent work (Rohrbach et al., 2017a) suggests that more meaningful captions can be achieved from an improved understanding of characters. In general, the ability to predict which character appears where and when facilitates a deeper understanding of videos that is grounded in the storyline.

Motivated by this goal, person clustering (Cinbis et al., 2011; Guillaumin et al., 2009; Jin et al., 2017; Tapaswi et al., 2019; Zhang et al., 2016b) and identification (Bäuml et al., 2013; Everingham et al., 2006; Nagrani and Zisserman, 2017; Ramanathan et al., 2014; Sivic et al., 2009) in videos has seen over a decade of research. In particular, fully automatic person identification is achieved in a weakly supervised manner either by aligning subtitles and transcripts (Bäuml et al., 2013; Everingham et al., 2006; Sivic et al., 2009), or using web images for actors and characters (Aljundi et al., 2016; Nagrani and Zisserman, 2017). On the other hand, clustering (Cinbis et al., 2011; Datta et al., 2018; Wu et al., 2013a,b; Zhang et al., 2016a,b) has mainly relied on *must-link* and *cannot-link* information obtained by tracking faces in a shot and analyzing their co-occurrence.

As face detectors improve (*e.g.* Hu and Ramanan (2017)), clustering and identification need to be revisited as more faces that exhibit extreme viewpoints, illumination, and resolution become available and need to be grouped or identified. Deep Convolutional Neural Networks (CNNs) have also yielded large performance gains for face representations (Cao et al., 2018; Parkhi et al., 2015; Schroff et al., 2015; Taigman et al., 2014). These networks are typically trained using hundreds-of-thousands to millions of face images gathered from the web, and show super-human performance on face verification tasks on images (LFW (Huang et al., 2008)) and videos (YouTubeFaces (Wolf et al., 2011)). Nevertheless, it is important to note that faces in videos such as TV series/movies exhibit more variety in comparison to *e.g.* LFW, where the images are obtained from Yahoo News by cropping mostly frontal faces. While these deep models generalize well, they are difficult to train from scratch (require lots of training data), and are typically transferred to other datasets via *net surgery*: fine-tuning (Sharma et al., 2018; Zhang et al., 2016a,b), or use of additional embeddings on the features from the last layer (Diba et al., 2017; Sarfraz et al., 2018; Tapaswi et al., 2019), or both.

Video face clustering also has potential applications in understanding other user-generated videos (*e.g.* content on YouTube) – mainly towards automatic summarization and content-based retrieval. For a method to work with such videos, it is especially important that the method be completely unsupervised (or self-supervised), as any required manual annotation will not scale with the exponential growth in the amount of video uploaded daily.

Representations. Clustering inherently builds on the notion of representations. We acknowledge the critical role of good features, and in this chapter, we address the problem of effectively learning representations to improve video face clustering. A good feature representation should exhibit small intra-person distances (*positive* pair of faces from the same person should be close) and large inter-person-distance (*negative* pair of faces from different people should be far). Recent works show that CNN representations can be improved via positive and negative pairs that are discovered through a Markov Random Field (MRF) (Zhang et al., 2016b); or a revised triplet-loss (Zhang et al., 2016a). In contrast, we propose methods that do not require complex optimization functions or supervision to improve the feature representation. We emphasize that while video-level constraints are not new, they need to be used properly to extract the most out of them. This is especially true in light of CNN face

representations that are very similar even across different identities. For example, Figure 3.5 shows a large overlap between the cosine similarity score distributions of positive (same identity) and negative (different identities) face pairs using base features. More importantly, the absolute values of similarity scores between different identities are surprisingly high and all above 0.93.

Contributions. Given a set of face images or tracks from several characters, our goal is to group them such that face images in a cluster belong to the same character. We propose and evaluate several simple ideas: discriminative and generative (see Section 3.3), that aim to further improve deep network representations. Note that all methods proposed in this chapter are either fully unsupervised, or use supervision that is obtained automatically, hence can be thought as unsupervised.

We propose two variants of *discriminative* approaches, and highlight the key differences below. In Track-supervised Siamese Network (TSiam), we include additional negative training pairs for singleton tracks – tracks that are not temporally co-occurring with any others in contrast to previous methods *e.g.* Cinbis et al. (2011). In our second approach, Self-supervised Siamese Network (SSiam), we obtain hard positive and negative pairs by sorting distances (*i.e.* ranking) on a subset of frames. Thus, SSiam can mine positive and negative pairs without the need for tracking, additionally enabling application of our method to image collections.

We compare our proposed methods against alternatives from *generative* modeling (auto-encoders) as strong baselines. In particular, Variational Autoencoder (VAE) (Kingma and Welling, 2014) can effectively model the distribution of face representations and achieve good generalization performance when working with the same set of characters. We perform extensive empirical studies and demonstrate the effectiveness and generalization of all methods. Our methods are powerful, yet simple, and obtain performance comparable or higher than state-of-the-art when evaluated on three challenging video face clustering datasets.

Further the chapter provides several key insights: (1) We discuss the use of generative models, in particular, Variational Autoencoders, as a strong baseline that can learn the latent identity information by modeling the underlying distribution of face representations. (2) We include an in depth empirical analysis and comparison of TSiam, SSiam, and VAE with comparison of generalization performance across videos

with same or different characters. (3) Finally, we include qualitative results to shed light on what the models may have learned as key identity information.

The remainder of this chapter is structured as follows: Section 3.2 provides an overview of related work. In Section 3.3, we propose TSiam, SSiam, and present a strong generative model as a baseline (VAE) to further refine deep features. Extensive experiments, an ablation study, comparison to the state-of-the-art, and qualitative results are presented in Section 3.4. We summarize key messages in a discussion (Section 3.5) and finally conclude in Section 3.6.

3.2 Related Work

Over the last decade, several advances have been made in video face clustering through discriminative models that aim to improve representations. In this section, we will review related work in this area, but also discuss some work on generative modeling (specifically VAEs) that may be used to improve face representations.

Generative face models. Along with MNIST handwritten digits (LeCun et al., 1998), faces are a common test bed for many generative models as they are a specific domain of images that can be modeled relatively well. Examples include Robust Boltzmann machines (Tang et al., 2012), and recent advances in Generative Adversarial Networks (GANs) that are able to generate stunning high resolution faces (Karras et al., 2018).

Variational Autoencoders (VAEs) have also seen growing use in face analysis, especially in generating new face images (Hou et al., 2017; Siddharth et al., 2017; Yan et al., 2016). In particular, (Yan et al., 2016) produces face images with desired attributes, while (Lindbo Larsen et al., 2015) combines VAEs and GANs towards the same goal. (Hou et al., 2017) replaces the pixel-level reconstruction loss by comparing similarity between deep representations. Recently, VAEs have been used to predict facial action coding (Tran et al., 2017) and model user reactions to movies (Deng et al., 2017).

There are some examples of VAEs adopted for clustering, however, video face datasets so far have not yet been considered. Stacked Autoencoders are used to simultaneously learn the representation and clustering (Xie et al., 2016), and Gaussian Mixture

Models are combined with VAEs for clustering (Dilokthanakul et al., 2016). Perhaps closest to our work, VAEs are used in conjunction with the triplet loss (Ishfaq et al., 2018) in a supervised way to learn good representations (but not evaluated on faces). We propose to use VAEs as a strong baseline and a different approach of learning feature representations as compared to standard discriminative approaches. To the best of our knowledge, we are the first to use VAEs in an unsupervised way to model and improve deep face representations, resulting in improved clustering performance.

Video face clustering. Clustering faces in videos commonly uses pairwise constraints obtained by analyzing tracks and some form of representation/metric learning. Different approaches can generally be categorized by the source of constraints.

One of the most commonly adopted source is the temporal information provided by face tracks. Face image pairs belonging to the same track are labeled positive (same character), while face images from co-occurring tracks help create negatives (different characters). This strategy has been exploited by learning a metric to obtain cast-specific distances (Cinbis et al., 2011) (ULDML); iteratively clustering and associating short sequences based on hidden Markov Random Field (HMRF) (Wu et al., 2013a,b); or performing clustering in a sub-space obtained by a weighted block-sparse low-rank representation (WBSLRR) (Xiao et al., 2014). In addition to pairwise constraints, video editing cues are used in an unsupervised way to merge tracks (Tapaswi et al., 2014b). Here, track and cluster representations are learned on-the-fly with dense-SIFT Fisher vectors (Parkhi et al., 2014). Recently, the problem of face detection and clustering is considered jointly (Jin et al., 2017), and a link-based clustering (Erdös-Rényi) based on rank-1 counts verification is adopted. The linking is done by comparing a given frame with a reference frame and learning a threshold to merge/not-merge frames.

Face track clustering/identification methods have also used additional cues such as clothing appearance (Tapaswi et al., 2012), speech (Paul et al., 2014), voice models (Nagrani and Zisserman, 2017), context (Zhang et al., 2013), gender (Zhou et al., 2015), name mentions (first, second, and third person references) in subtitles (Haurilet et al., 2016), multispectral information (Sharma and Van Gool, 2016; Sharma et al., 2016), pose information (Sarfraz and Hellwich, 2008a,b, 2010), weak labels using transcripts/subtitles (Bäuml et al., 2013; Everingham et al., 2006), and joint action and actor labeling (Miech et al., 2017a) using transcripts.

With the popularity of CNNs, there is a growing focus on improving face representations using video-level constraints. An improved form of triplet loss is used to fine-tune the network and push the positive and negative samples apart in addition to requiring anchor and positive to be close, and anchor and negative far (Zhang et al., 2016a). Zhang et al. (2016b) learn better representations by dynamic clustering constraints that are discovered iteratively during clustering that is performed via a Markov Random Field (MRF). Roethlingshoefer et al. (2019) use graph neural network to learn representation of face-tracks. In contrast to related work, we propose a simple, yet effective approach (SSiam) to learn good representations by sorting distances on a subset of frames and not requiring video/track level constraints to generate positive/negative training pairs.

Another point of comparison lies in Zhang et al. (2016a,b) and Datta et al. (2018) who only use video-level constraints to generate a set of similar and dissimilar face pairs. Thus, the model does not see negative pairs for singleton (non co-occurring) tracks. In contrast, our method TSiam incorporates negative pairs for the singleton tracks by exploiting track-level distances.

Recently, advances in clustering approaches themselves have contributed to better performance. Sarfraz et al. (2019) propose a new clustering algorithm (FINCH) based on first neighbor relations. However, FINCH is not trainable in contrast to our method, and would only benefit further from improved feature representations. In He et al. (2018), the authors use inverse reinforcement learning on a ground-truth dataset to find a reward function for deciding whether to merge a given pair of facial features. Contrary to these methods, we expect neither the existence of ground-truth data, nor a measure of face quality. Parallel to this work, Tapaswi et al. (2019) propose to learn an embedding space that creates a fixed-radius ball for each character thus allowing to estimate the number of clusters. However, their work requires supervised labels during training, while our models learn the embedding in a self-supervised setting.

Finally, there are related works that "harvest" training data from unlabeled sources which is in the similar spirit of SSiam and TSiam. Fernando et al. (2017) and Misra et al. (2016) shuffle the video frames and treat them as positive or negative training data for reordering video frames; Wang and Gupta (2015) collect positive and negative training data by tracking bounding boxes (*i.e.* motion information) in

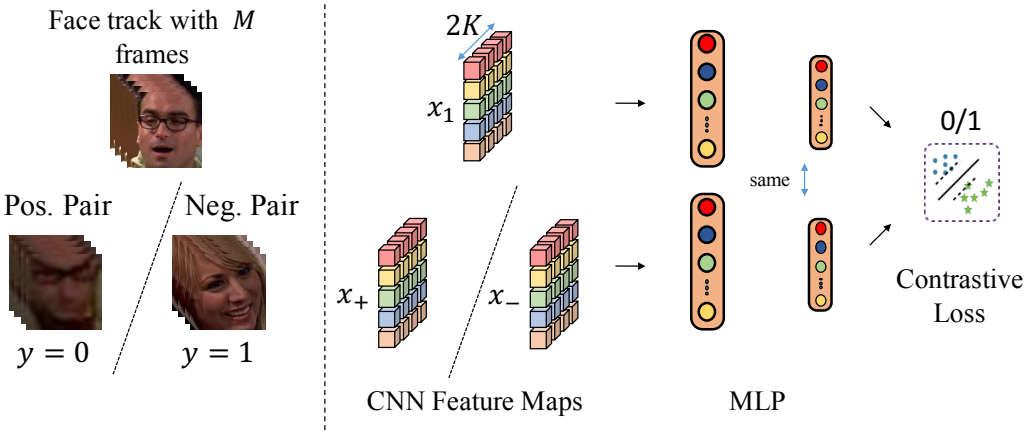


Figure 3.1: Track-supervised Siamese network (TSiam). Illustration of the Siamese architecture used in our track-supervised Siamese networks. Note that the MLP is shared across both feature maps. $2K$ corresponds to batch size.

order to learn effective visual representations. In contrast, we propose new techniques to generate labels and utilize them efficiently to improve video face clustering.

3.3 Refining Face Representations for Clustering

Our goal is to improve face representations using simple methods that build upon the success of deep CNNs. More precisely, we propose models to refine the face descriptors automatically, without the need for manually curated labels. Note that we do not fine-tune the base CNN, and only learn a few linear layers above it. Our approach has three key benefits: (i) it is easily applicable to new videos (does not require labels); (ii) it does not need large amounts of training data (few hundred tracks are enough); and (iii) specialized networks can be trained to specialize on each episode or film.

We start this section by first introducing the notation used throughout the remainder of the chapter. We then propose the discriminative models: (1) Track-supervised Siamese Network (TSiam), and (2) Self-supervised Siamese Network (SSiam) (Section 3.3.1). Finally, we present how Variational Autoencoders (VAE) can be used to improve representation learning and act as a strong generative model baseline (Section 3.3.2).

Preliminaries. Consider a video with N face tracks $\{T^1, \ldots, T^N\}$ belonging to C characters. Each track corresponds to one of the characters, and consists of

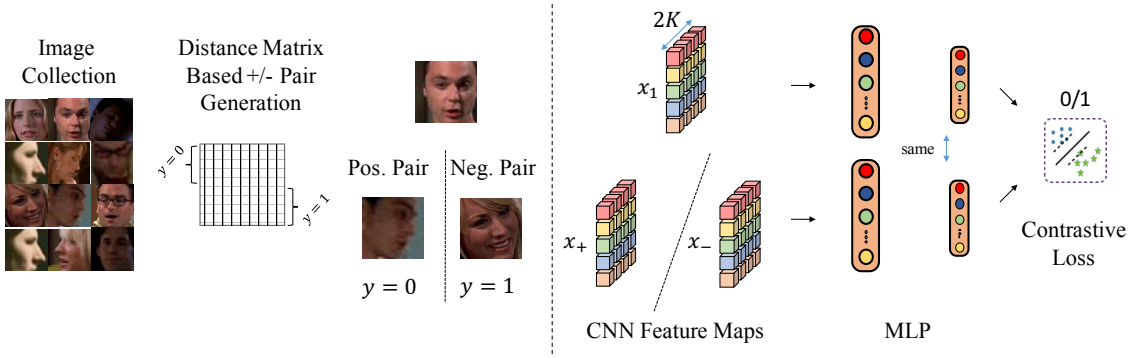


Figure 3.2: Self-supervised Siamese network (SSiam). Illustration of the Siamese architecture used in our self-supervised Siamese networks. SSiam selects hard pairs: farthest positives and closest negatives using a ranked list based on Euclidean distance for learning similarity and dissimilarity respectively. Note that the MLP is the same across both feature maps. $2K$ corresponds to batch.

$T^i = \{f_1, \ldots, f_{M^i}\}$ face images. Our goal is to group tracks into sets $\{G_1, \ldots, G_{|C|}\}$ such that each track is assigned to only one group, and ideally, each group contains all tracks from the same character. We use a deep CNN (VGG2 (Cao et al., 2018)) and extract a descriptor for each face image $\mathbf{x}_k^i \in \mathbb{R}^D$, $k = 1, \ldots, M^i$ from the penultimate layer (before classification) of the network. We refer to these as *base features*, and demonstrate that they already achieve a high performance. As a form of data augmentation, we use 10 crops obtained from an expanded bounding box surrounding the face image during training. Evaluation is based on one center crop.

Track-level representations are obtained by aggregating the face image descriptors

$$\mathbf{t}^i = \frac{1}{M^i} \sum_k \mathbf{x}_k^i \,. \tag{3.1}$$

We additionally normalize track representations to be unit-norm, $\hat{\mathbf{t}}^i = \mathbf{t}^i / \|\mathbf{t}^i\|_2$ before using them for clustering.

Hierarchical Agglomerative Clustering (HAC) has been the clustering method adopted by several previous works (Tapaswi et al., 2014b; Zhang et al., 2016a,b). For a fair comparison, we also use HAC to obtain a fixed number of clusters equal to the number of characters (known a priori). We use the minimum variance ward linkage (Ward Jr., 1963) for all methods which is nowadays commonly used for HAC. See Figure 3.4 for an illustration.

3.3.1 DISCRIMINATIVE MODELS

Discriminative clustering models typically associate a binary label y with a pair of features. We designate $y = 0$ when a pair of features $(\mathbf{x}_1, \mathbf{x}_2)$ belong to the same character (identity), and $y = 1$ otherwise (Hadsell et al., 2006).

We use a shallow MLP to reduce the dimensionality and improve generalization of the features (see Figure 3.1, 3.2). Here, each face image is encoded as $Q_\phi(\mathbf{x}_k^i)$, where ϕ corresponds to the trainable parameters of the MLP. We find $Q_\phi(\cdot)$ to perform best when using a linear layer (for details see Section 3.4.2). To perform clustering, we compute track-level aggregated features by average pooling across the embedded frame-level representations (Sharma et al., 2017)

$$\mathbf{t}^i = \frac{1}{M^i} \sum_k Q_\phi(\mathbf{x}_k^i)\,, \tag{3.2}$$

followed by ℓ_2-normalization.

We train our model parameters by minimizing the contrastive loss (Hadsell et al., 2006) at the frame-level:

$$\begin{aligned} &\mathcal{L}\left(W, y, Q_\phi(\mathbf{x}_1), Q_\phi(\mathbf{x}_2)\right) = \\ &\quad \frac{1}{2}\left((1-y)\cdot(d_W)^2 + y\cdot(\max(0, m - d_W))^2\right)\,, \end{aligned} \tag{3.3}$$

where $\mathbf{x}_1$ and $\mathbf{x}_2$ are a pair of face representations with $y = 0$ when coming from the same character, and $y = 1$ otherwise. $W : \mathbb{R}^{D\times d}$ is a linear layer that embeds $Q_\phi(\mathbf{x})$ such that $d \ll D$ (in our case, $d = 2$). d_W is the Euclidean distance $d_W = \|W \cdot Q_\phi(\mathbf{x}_1) - W \cdot Q_\phi(\mathbf{x}_2)\|^2$, and m is the margin, empirically chosen to be 1.

In the following, we present two strategies to automatically obtain supervision for pairs of frames: Figure 3.1 illustrates the Track-level supervision, and Figure 3.2 shows the Self-supervision for Siamese network training.

Track-supervised Siamese network (TSiam).

Video face clustering often employs face tracking to link face detections made in a series of consecutive frames. The tracking acts as a form of high precision clustering

(grouping detections within a shot) and is popularly used to automatically generate positive and negative pairs of face images (Cinbis et al., 2011; Datta et al., 2018; Tapaswi et al., 2014b; Wu et al., 2013b). In each frame, we assume that characters appear on screen only once. Thus, all face images within a track can be used as positive pairs, while face images from co-occurring tracks are used as negative pairs. For each frame in the track, we sample two frames within the same track to form positive pairs, and sample four frames from a co-occurring track (if it exists) to form negative pairs.

Depending on the filming style of the series/movie, characters may appear alone or together on screen. As we will see through experiments on diverse datasets, some videos have 35% tracks with co-occurring tracks, while this can be as large as 70% for other videos. For isolated tracks, we sort all other tracks in the same video based on track-level distances (computed on base features) and randomly sample frames from the farthest $F = 25$ tracks. Note that all previous works ignore negative pairs for singleton (not co-occurring) tracks. We will highlight their impact in our experiments.

Self-supervised Siamese network (SSiam)

Supervision from tracking may not always be available or may also be unreliable. An example is face clustering within image collections (*e.g.* on social media platforms). To enable the use of metric learning without any supervision we propose an effective approach that can generate the required pairs automatically during training. SSiam is inspired by pseudo-relevance feedback (pseudo-RF) (Yan et al., 2003a,b) that is commonly used in information retrieval.

We hypothesize that the first and last samples of a ranked list based on Euclidean distance are strong candidates for learning similarity and dissimilarity respectively. We exploit this in a meaningful way and generate promising similar and dissimilar pairs from a representative subset of the data.

Formally, consider a subset $\mathcal{S} = \{\mathbf{x}_1, \ldots, \mathbf{x}_B\}$ of face image representations from the dataset (sampled randomly, not from the same track). We treat each frame $\mathbf{x}_b, b = 1, \ldots, B$ as a query and compute Euclidean distance against every other frame in the set. We sort rows of the resulting matrix in an ascending order (smallest to

largest distance) to obtain an ordered index matrix $\mathcal{O}(\mathcal{S}) = [s_1^o; \ldots; s_B^o]$. Each row s_b^o contains an ordered index of the closest to farthest faces corresponding to $\mathbf{x}_b$. Note that the first column of such a matrix is the index b itself at distance 0. The second column corresponds to nearest neighbors for each frame and can be used to form the set of positive pairs $\mathcal{S}_+$. Similarly, the last column corresponds to farthest neighbors and forms the set of negative pairs $\mathcal{S}_-$. Each element of the above sets stores: query index b, nearest/farthest neighbor r, and the Euclidean distance d.

During training, we first form pairs dynamically by picking a random subset of B frames at each iteration. We compute the distances, sort them, and obtain positive and negative pairs sets $\mathcal{S}_+, \mathcal{S}_-$, each with B elements as described above. Among them, we choose K pairs from the positive set that have the largest distances and K pairs from the negative set with the smallest distances. This allows us to select semi-hard positive pairs and semi-hard negative pairs from each representative set of B elements. Finally, these $2K$ pairs are used in a contrastive setting (Eq. 3.3) to train network parameters.

To encourage variety in the sample set $\mathcal{S}$ and reduce the chance of false positives/negatives in the chosen $2K$ pairs, B is chosen to be much larger than K ($B = 1000, K = 64$). Experiments on several datasets and generalization studies show the benefit and effectiveness of this approach in collecting positive and negative pairs to train the network.

Note that, SSiam can be thought of as an improved version of pseudo-RF with batch processing. Rather than selecting farthest negatives and closest positives for each independent query, we emphasize that SSiam selects $2K$ hard pairs: farthest positives and closest negatives by looking at the batch of queries B jointly. This selection of sorted pairs from the positive $\mathcal{S}_+$ and negative $\mathcal{S}_-$ sets is quite important as will be shown later.

3.3.2 GENERATIVE MODELS

We now present generative models as an alternative strong baseline that can also achieve similar improvements to feature representations. Similar to SSiam, we do not require track-level supervision, in fact, auto-encoders consider single images (and not pairs) at a time.

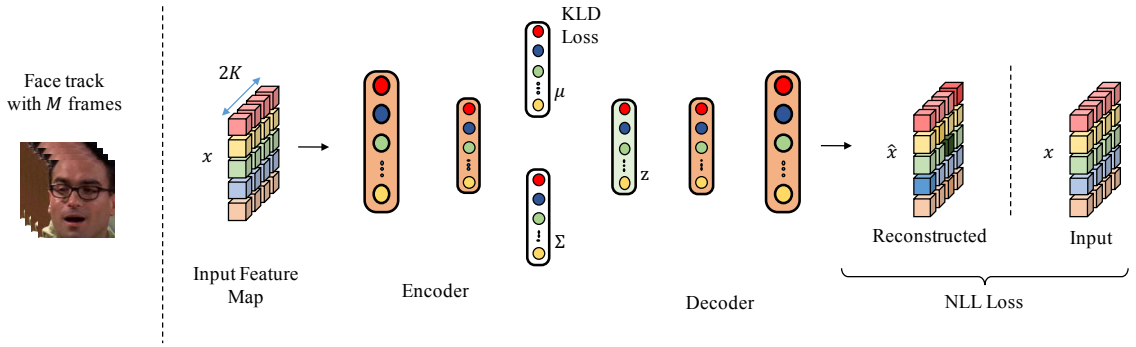


Figure 3.3: Illustration of a Variational Autoencoder used as a strong baseline generative model. In contrast to the Siamese networks, the VAE sees single frames (not pairs) and is trained by two losses: KL-Divergence and the Reconstruction NLL. $2K$ corresponds to batch.

Variational Autoencoder (VAE)

Deep face CNNs are trained to identify and distinguish between people, and are supposed to be invariant to effects of pose, illumination, *etc.*. However, in reality, pose, background, and other image-specific characteristics leak into the model, reducing performance. Our goal is to learn a latent variable model that separates identity from other spurious artifacts given a deep representation. We assume that face descriptors are generated by a random process that first involves sampling a continuous latent variable $\boldsymbol{z}$ representing identity of the characters. This is followed by a conditional model $p(\mathbf{x}|\boldsymbol{z};\theta)$ with some parameters θ, modeled as a neural network (specifically, an MLP)

$$p(\mathbf{x}) = \int p(\mathbf{x}|\boldsymbol{z};\theta)p(\boldsymbol{z})d\boldsymbol{z}\,. \tag{3.4}$$

We propose to adopt a Variational Autoencoder (VAE) (Kingma and Welling, 2014) consisting of an encoder MLP $Q(\boldsymbol{z}|\mathbf{x};\phi)$ with parameters ϕ and the decoder $P(\mathbf{x}|\boldsymbol{z};\theta)$. The encoder provides an approximate posterior over the latent variable, and the model parameters are trained to maximize the variational lower bound

$$\mathcal{L}_v = \mathbb{E}_{q(\boldsymbol{z}|\mathbf{x};\phi)}[\log p(\mathbf{x}|\boldsymbol{z};\theta)] - D_{KL}(q(\boldsymbol{z}|\mathbf{x};\phi)\|p(\boldsymbol{z}))\,. \tag{3.5}$$

Note that $\log p(\mathbf{x}) \geq \mathcal{L}_v$ and thus maximizing $\mathcal{L}_v$ corresponds to maximizing the log-likelihood of the samples. D_{KL} is the Kullback-Leibler Divergence between the approximate posterior $q(\boldsymbol{z}|\mathbf{x};\phi)$ and the latent variable prior $p(\boldsymbol{z})$, and acts like a regularization on the distribution of latent variables. The first term corresponds to the

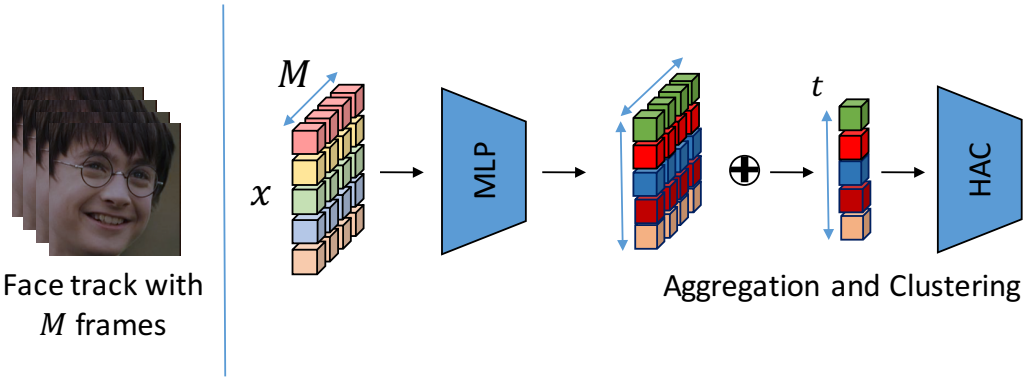


Figure 3.4: Illustration of the test time evaluation scheme. Given our pre-trained MLPs, TSiam or SSiam, we extract the frame-level features for the track, followed by mean pooling to obtain a track-level representation. All such track representations from the video are grouped using HAC to obtain a known number of clusters.

log-likelihood of observing $\mathbf{x}$ given $\boldsymbol{z}$ and is a form of reconstruction error. Note that it requires sampling from $q(\boldsymbol{z}|\mathbf{x};\phi)$ which is achieved using the reparameterization trick (Kingma and Welling, 2014). In practice, for each input $\mathbf{x}$, the encoder MLP Q_ϕ predicts the latent variable mean $\boldsymbol{\mu}$ and a diagonal variance $\boldsymbol{\Sigma}$ that model a Gaussian prior. We refer the interested reader to Doersch (2016) for a gentle introduction.

Our encoder is a two-layer MLP Q_ϕ and generates $(\boldsymbol{\mu}_k^i, \boldsymbol{\Sigma}_k^i)$ for each face image representation $\mathbf{x}_k^i$. The decoder is also a two-layer MLP P_θ that takes as input a latent variable sample

$$\boldsymbol{z}_k^i = \boldsymbol{\mu}_k^i + \epsilon {\boldsymbol{\Sigma}_k^i}^{0.5}, \quad \epsilon \sim \mathcal{N}(0, I)\,, \tag{3.6}$$

and produces a reconstruction $\hat{\mathbf{x}}_k^i = P_\theta(\boldsymbol{z}_k^i)$. Both the reconstructed representation $\hat{\mathbf{x}}_k^i$ and the latent variable predicted mean $\boldsymbol{\mu}_k^i$ can be used for clustering. We form two final track representations based on the reconstructed features $\mathbf{t}_{rec}^i = \frac{1}{M^i}\sum_k \hat{\mathbf{x}}_k^i$ and based on the latent means $\mathbf{t}_{\boldsymbol{\mu}}^i = \frac{1}{M^i}\sum_k \boldsymbol{\mu}_k^i$. Figure 3.3 illustrates the model.

Note that, we propose VAEs as an alternative method to discriminative approaches, and a strong baseline. We show in our experiments that discriminative methods TSiam and SSiam, often perform equally or better than VAEs.

3.4 Evaluation

We present our evaluation on three challenging datasets. We first describe the clustering metric, followed by a thorough analysis of the proposed methods, ending with a comparison to state-of-the-art.

3.4.1 Experimental Setup

Datasets and metric. We present our evaluation on three challenging datasets: BBT, BF and ACCIO, discussed in Section 2.3. We summarize the statistics of datasets in Table 2.1. In terms of evaluation metric following the previous works, for BBT and BF we report clustering accuracy (ACC or WCP), and for ACCIO in addition to ACC, we report BCubed Precision (P), Recall (R) and F-measure (F), as discussed in Section 2.4.

3.4.2 Implementation Details

Figure 3.4 illustrates the network architecture during test time.

CNN. We adopt the VGG-2 face CNN (Cao et al., 2018), a ResNet50 model, pre-trained on MS-Celeb-1M (Guo et al., 2016) and fine-tuned on 3.31 million face images of 9,131 subjects (VGG2 data). Input RGB face images are resized to 224×224, and pushed through the CNN. We extract `pool5_7x7_s1` features, resulting in $\mathbf{x}_k^i \in \mathbb{R}^{2048}$.

Siamese network MLP. The network[1] comprises of two fully-connected layers ($\mathbb{R}^{2048} \rightarrow \mathbb{R}^{256} \rightarrow \mathbb{R}^2$). Note that the second linear layer is part of the contrastive loss (corresponds to W in Eq. 3.3), and we use the feature representations at $\mathbb{R}^{256}$ for clustering.

[1]We optimized our MLP network architecture by varying the number of hidden layers, and the number of units in each layer. We varied the number of layers between one-layer to three-layer model with number of units varying between {256,512,1024} (first layer), {2,64,128,256,512} (second layer) and {2,64,128,256,512} (third layer). We found that our two-layer model ($\mathbb{R}^{256} \rightarrow \mathbb{R}^2$) and the extracted feature representations from the first layer (feat-dim=256) generalized over all datasets and give the best performance.

We train our Siamese network with track-level supervision (TSiam) with about 102k positive and 204k negative frame pairs (for BBT-0101) by mining 2 positive and 4 negative pairs for each frame. For the Self-supervised Siamese network (SSiam), we generate batches of size $B = 1000$, and select $K = 64$ positive and negative pairs each. Higher batch sizes $B = 2000, 3000$, did not provide significant improvements.

The MLP is trained using the contrastive loss, and parameters are updated using Stochastic Gradient Descent (SGD) with a fixed learning rate of 10^{-3}. Since the labels are obtained automatically for each video, overfitting is not a concern. We train our model until convergence (loss does not reduce significantly any further).

VAE. Our VAE[2] uses a two-layer MLP encoder ($\mathbb{R}^{2048} \rightarrow \mathbb{R}^{1024} \rightarrow \mathbb{R}^{256\times2}$, $\boldsymbol{\mu}$ and diagonal co-variance $\boldsymbol{\Sigma}$); and a two-layer MLP decoder ($\mathbb{R}^{256} \rightarrow \mathbb{R}^{1024} \rightarrow \mathbb{R}^{2048}$). The VAE is trained using SGD, with a learning rate of 10^{-3} until convergence.

3.4.3 CLUSTERING PERFORMANCE ABLATION STUDIES

Table 3.1: Clustering accuracy on the base face representations.

Dataset	Track-level		Frame-level	
	VGG1	VGG2	VGG1	VGG2
BBT-0101	0.916	**0.932**	0.938	**0.940**
BF-0502	0.831	**0.836**	0.901	**0.912**

Base features. We begin our analysis by comparing track- and frame-level performance of two commonly used CNNs to obtain face representations: VGG1 (Parkhi et al., 2015) and VGG2 (Cao et al., 2018). Track-level results use mean-pool of frames. Results are reported in Table 3.1. Note that the differences between VGG1 and VGG2 are typically within 1% of each other indicating that the results in the subsequent experiments are not just due to having better CNNs trained with more data. We refer to VGG2 features as Base for the remainder of this chapter.

[2]From our MLP architecture search, we exploited a lot of insights and restricted our VAE architecture to two-layer. We optimized our VAE network architecture by varying the number number of units of first linear layer ({256,512,1024}), $\boldsymbol{\mu}$ ({64,128,256}), and $\boldsymbol{\Sigma}$ ({64,128,256}). We found that the reconstructed feature representation generalized over all datasets and give the best performance.

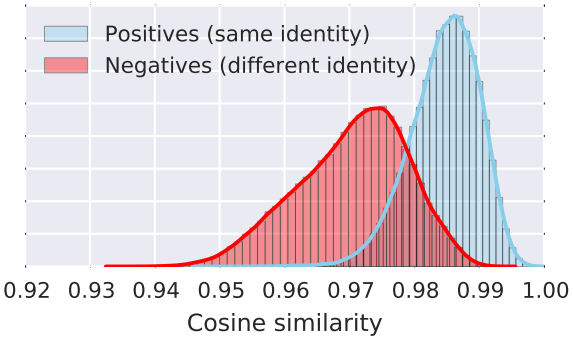


Figure 3.5: Histograms of pairwise cosine similarity between tracks of same identity (positive, blue) and different identity (negative, red) for BBT-0101. Best seen in color.

Role of effective mining of +/- pairs. We emphasize that especially in light of CNN face representations, the features are very similar even across different identities, and thus positive and negative pairs need to be created carefully to gain the most improvements. Figure 3.5 proves this point as (i) we see a large overlap between the cosine similarity distributions of positive (same identity) and negative (different identities) track pairs on the base features; and (ii) note the scale on the x-axis, even negative pairs have cosine similarity scores higher than 0.9.

TSiam, impact of singleton tracks. Previous work with video-level constraints (Zhang et al., 2016a,b) and (Datta et al., 2018), ignore singleton (not co-occurring) tracks. In TSiam, we include negative pairs for singletons based on track distances. Table 3.2 shows that 50-70% tracks are singleton and ignoring them lowers accuracy by 3-4%. This confirms our hypothesis that incorporating negative pairs of singletons helps improve performance.

Table 3.2: Ignoring singleton tracks (and possibly characters) leads to significant performance drop. Accuracy on track-level clustering.

	TSiam		# Tracks		
Dataset	w/o Single (Datta et al., 2018)	Ours	Total	Single	Co-oc
BBT-0101	0.936	**0.964**	644	331	313
BF-0502	0.849	**0.893**	568	395	173

SSiam and Pseudo-Relevance Feedback. In Pseudo-RF (Yan et al., 2003a,b), all samples are treated independent of each other, there is no batch of data B from which $2K$ pairs are chosen. A pair of samples closest in distance are chosen as positive, and farthest as negative. However, this usually corresponds to samples

that already satisfy the loss margin, thus leading to small (possibly even 0) gradient updates. Table 3.3 shows that SSiam that involves sorting a batch of queries is much more effective than pseudo-RF as it has the potential to select harder positives and negatives. We see a consistent gain in performance, 3% for BBT-0101 and over 9% for BF-0502.

Table 3.3: Comparison between SSiam and pseudo-RF.

Method	BBT-0101	BF-0502
Pseudo-RF	0.930	0.814
SSiam	**0.962**	**0.909**

3.4.4 STUDYING GENERALIZATION

Please note that generalization experiments are presented here to explore the underlying properties of our discriminative and generative models. If achieving high performance is the only goal, we assert that our models can be trained and evaluated on each video rapidly and fully automatically.

Performance on training videos. We report clustering performance on training videos in Table 3.4. Note that all our models are trained in an unsupervised manner, or with automatically generated labels. We additionally report results for an AutoEncoder (AE) with the same network architecture as the VAE, but without the variational space and sampling. The AE is trained using NLL loss for reconstruction.

We observe that VAE and SSiam show large performance boost over the base VGG2 features on BBT and BF. In particular, VAE shows large improvement on videos with few characters (BBT). With training and evaluation on the same video, the generative models demonstrate comparable performance to discriminative models.

Table 3.4: Clustering accuracy computed at track-level on the training episodes, with a comparison to all evaluated models.

Train/Test	Base	TSiam	SSiam	AE	VAE
BBT-0101	0.932	0.964	0.962	0.967	**0.984**
BF-0502	0.836	0.893	**0.909**	0.842	0.889

Generalization within series. In this experiment, we evaluate the generalization capability of our models. We train on one episode each, BBT-0101 and BF-0502, and evaluate on all other episodes of the same TV series. Table 3.5 reports averaged clustering accuracy over the remaining 5 episodes for each series. Both SSiam or TSiam perform similar (slightly lower/higher) to the base features, possibly due to overfitting. VAE performs better here in comparison to the discriminative models. This indicates that VAEs are able to model the distribution of face representations by extracting the latent character identity that is common across the episodes.

Table 3.5: Clustering accuracy computed at track-level across episodes within the same TV series. Numbers are averaged across 5 test episodes.

Train	Test	Base	TSiam	SSiam	AE	VAE
BBT-0101	BBT-01[02-06]	0.935	0.930	0.914	0.917	**0.945**
BF-0502	BF-05[01,03-06]	0.892	0.889	0.904	0.899	**0.908**

Generalization across series. We further analyze our models by evaluating generalization across series. Based on Table 3.6, we bring the readers attention towards three key observations:

1. TSiam and SSiam retain their discriminative power and can transfer to other series more gently. As they learn to score similarity between pairs of faces, the underlying distribution of identities does not matter much. For example, the drop when training TSiam on BBT-0101 and evaluating on BF is 0.890 (train on BF-0502) to 0.875.

2. As filming styles differ, underlying distributions of the face identities can be quite different. VAEs are unable to cope with this shift, and show drop in performance. Training on BBT-0101 and evaluating on BF reduces performance from 0.905 (train on BF-0502) to 0.831.

3. We clearly see that the similarity between videos can affect generative models. For example, BBT is quite similar with mostly bright scenes during the day. BF on the other hand has almost half the scenes at night causing large variations.

Generalization to unseen characters. In the ideal setting, we would like to cluster all characters appearing in an episode including the main characters, other named characters, and the unknown characters. However, this is a very difficult

Table 3.6: Clustering accuracy when evaluating across video series. Each row indicates that the model was trained on one episode of BBT / BF, but evaluated on all 6 episodes of the two series.

	Train Episode	Test series BBT-01[01-06]	Test series BF-05[01-06]
TSiam	BBT-0101	0.936	0.875
	BF-0502	0.915	0.890
SSiam	BBT-0101	0.922	0.862
	BF-0502	0.883	0.905
VAE	BBT-0101	0.952	0.831
	BF-0502	0.830	0.905

setting, and in fact, disambiguating background characters is even hard for humans and there are no datasets that include such labels. For BBT and BF, we do however have all named characters labeled. Firstly, expanding the clustering experiment to include them drastically changes the class balance. For example, BF-0502 has 6 main and 12 secondary characters with class balance shifting from 36.2/5.0 to 40.8/0.1 (lowest to highest cluster membership in percentage).

We present clustering accuracy for this setting in Table 3.7. All proposed methods show a drop in performance when extending to unseen characters. Note that the models have been trained on only the main characters data and tested on all (including unseen) characters. However, the drop is small when adding just 1 new character (BBT-0101) vs. introduction of 6 in BF-0502.

SSiam's performance generalizes gracefully, probably since it is trained with a diverse set of pairs (dynamically generated during training) and can generalize to unseen characters.

Table 3.7: Clustering accuracy when extending to all named characters within the episode. BBT-0101 has 5 main and 6 named characters. BF-0502 has 6 main and 12 named characters.

	BBT-0101			BF-0502		
	TSiam	SSiam	VAE	TSiam	SSiam	VAE
Main cast	0.964	0.962	0.984	0.893	0.909	0.889
All named cast	0.958	0.922	0.978	0.829	0.870	0.807

Number of clusters when purity = 1. Table 3.8 shows the number of clusters we can achieve while maintaining purity to be 1. In a similar spirit to Tapaswi et al. (2014b), this metric indicates when the first mistake in agglomerative merging occurs – smaller the number, the better it is. SSiam works best on the harder BF dataset, while VAE can reduce the clusters most on BBT.

Table 3.8: In a similar spirit to Tapaswi et al. (2014b), we evaluate the number of clusters we can reach when maintaining clustering accuracy/purity at 1. Lower is better.

Video	#Tracks	Base	TSiam	SSiam	VAE	Ideal
BBT-0101	644	365	369	389	**245**	5
BF-0502	568	460	490	**253**	312	6

Generalization to joint training. For this evaluation, we train a model combining BBT-0101, BF-0502, and NH [3]. We report results in Table 3.9. The drop in performance is expected, however, note that unsupervised overfitting to each episode is not necessarily bad. Interestingly, VAE retains performance on BF-0502, we suspect this may be due to more visual variation in BF. Our discriminative methods do perform well when trained and evaluated on larger datasets (see Table 3.11), while VAE suffers due to large number of characters and a high skew in cluster ratios.

We show generalization studies to better understand our methods. VAEs seem to transfer well to within domain (same characters), while discriminative TSiam and SSiam transfer well across TV series. However, training on each episode should yield best performance for all methods.

Table 3.9: Impact of training on combined dataset of BBT-0101, BF-0502, and NH.

	Train	TSiam	SSiam	VAE
BBT-0101	BBT-0101	**0.964**	**0.962**	**0.984**
	BBT+BF+NH	0.930	0.930	0.938
BF-0502	BF-0502	**0.893**	**0.909**	0.889
	BBT+BF+NH	0.852	0.887	**0.890**

[3] *Notting Hill* (NH) (Wu et al., 2013b; Zhang et al., 2016b): a romantic comedy movie. NH has 5 main casts with 240 tracks, and 16872 frames. The LC/SC (%) is 43.1/7.4. Tracks for NH are provided by (Wu et al., 2013b).

3.4.5 COMPARISON WITH THE STATE-OF-THE-ART

BBT and BF. We compare our proposed methods (TSiam, and SSiam) with the state-of-the-art approaches in Table 3.10. We report clustering accuracy (%) on two videos: BBT-0101 and BF-0502. Historically, previous works have reported performance at a frame-level. We follow this for TSiam and SSiam.

Unlike previous works (Zhang et al., 2016a,b) that use face-tracks from Cinbis et al. (2011); Everingham et al. (2006); Roth et al. (2012), we use state-of-the-art face detection and tracking algorithms that recognizes and tracks faces even in poor illumination and pose variations, thus resulting in a larger number of frames and face-tracks. Note that our evaluation uses 2-4 times larger number of frames than previous works (Zhang et al., 2016a,b) making direct comparison hard. Specifically in BBT-0101 we have 41,220 frames while Zhang et al. (2016a) uses 11,525 frames. Similarly, we use 39,263 frames for BF-0502 (vs. 17,337 (Zhang et al., 2016b)). We use these data sources for evaluation of BBT and BF throughout the thesis. Even though we cluster more frames and tracks (with more visual diversity), our approaches are comparable to or even better than the current results.

TSiam, SSiam and VAE are all better than the improved triplet method (Zhang et al., 2016a) on BBT-0101. SSiam obtains 99.04% accuracy which is 3.04% higher, and VAE obtains 2.4% better performance (absolute gains). On BF-0502, TSiam performs the best with 92.46% which is 0.33% better than the JFAC (Zhang et al., 2016b).

ACCIO. We also evaluate our methods on the ACCIO dataset with 35 named characters, 3,243 tracks, and 166,885 faces[4]. The largest to smallest cluster ratios are very skewed: 30.65% and 0.06%. In fact, half the characters correspond to less than 10% of all tracks. Table 3.11 presents the results when performing clustering to yield 36 clusters (equivalent to the number of characters). In addition, as in Zhang et al. (2016b), Table 3.12 (num. clusters = 40) shows that our discriminative methods are not affected much by this skew, and in fact improve performance by a significant margin over the state-of-the-art.

[4]Note that, in accordance with previous literature (Zhang et al., 2016b), we use 36 and 40 clusters for the 35 main characters.

Table 3.10: Comparison to state-of-the-art. Metric is clustering accuracy (%) evaluated at frame-level. Please note that many previous works use fewer tracks (# of frames) (also indicated in Table 2.1) making the task relatively easier. We use an updated version of face tracks provided by Bäuml et al. (2013).

Method	Data Source			
	BBT-0101	BF-0502	BBT	BF
ULDML (ICCV '11) (Cinbis et al., 2011)	57.00	41.62	–	Cinbis et al. (2011)
HMRF (CVPR '13) (Wu et al., 2013b)	59.61	50.30	Roth et al. (2012)	Everingham et al. (2006)
HMRF2 (ICCV '13) (Wu et al., 2013a)	66.77	–	Roth et al. (2012)	–
WBSLRR (ECCV '14) (Xiao et al., 2014)	72.00	62.76	–	Everingham et al. (2006)
VDF (CVPR '17) (Sharma et al., 2017)	89.62	87.46	Bäuml et al. (2013)	Bäuml et al. (2013)
Imp-Triplet (PacRim '16) (Zhang et al., 2016a)	96.00	–	Roth et al. (2012)	–
JFAC (ECCV '16) (Zhang et al., 2016b)	–	92.13	–	Everingham et al. (2006)
TSiam	**98.58**	**92.46**		
SSiam	**99.04**	90.87	Bäuml et al. (2013)*	Bäuml et al. (2013)*
VAE	**98.40**	85.30		

Table 3.11: Performance comparison of TSiam and SSiam with JFAC (Zhang et al., 2016b) on ACCIO with 36 clusters.

Methods	#cluster=36 P	R	F
JFAC (ECCV '16) (Zhang et al., 2016b)	0.690	0.350	0.460
TSiam	**0.749**	**0.382**	**0.506**
SSiam	**0.766**	**0.386**	**0.514**
VAE	**0.710**	0.325	0.446

Table 3.12: Performance comparison of different methods on ACCIO with 40 clusters.

Methods	# clusters=40 P	R	F
DIFFRAC-DeepID2$^+$ (ICCV '11) (Zhang et al., 2016b)	0.557	0.213	0.301
WBSLRR-DeepID2$^+$ (ECCV '14) (Zhang et al., 2016b)	0.502	0.206	0.292
HMRF-DeepID2$^+$ (CVPR '13) (Zhang et al., 2016b)	0.599	0.230	0.332
JFAC (ECCV '16) (Zhang et al., 2016b)	0.711	0.352	0.471
TSiam	**0.763**	**0.362**	**0.491**
SSiam	**0.777**	**0.371**	**0.502**
VAE	**0.718**	0.305	0.428

Computational complexity. Our models essentially consist of a few linear layers and are very fast to compute at inference time. In fact, training the SSiam for about 15 epochs on BBT-0101 requires less than 25 minutes (on a GTX 1080 GPU using the matconvnet framework (Vedaldi and Lenc, 2015)).

3.5 DISCUSSION

Comparison of TSiam/SSiam to VAE. In Table 3.13, we report the frame-level clustering performance of both discriminative (*i.e.* TSiam and SSiam) and generative (VAE) methods. We observe that TSiam and SSiam generally perform better than the base features. In addition, SSiam is generally better than TSiam. VAEs exhibit an unclear fluctuating behavior, and in fact, have worse performance than base features on harder datasets (BF, ACCIO).

Table 3.13: Clustering accuracy (%) performance comparison of TSiam, SSiam, and VAE over all three dataset BBT-0101, BF-0502 and ACCIO

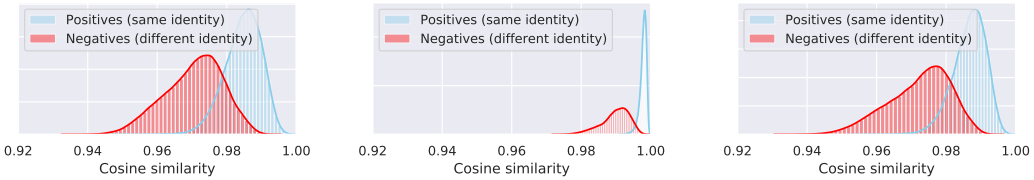
	#Clusters	Base	TSiam	SSiam	VAE
BBT-0101	5	94.00	98.58	**99.04**	98.40
BF-0502	6	91.20	**92.46**	90.87	85.30
ACCIO	36	79.90	81.30	**82.00**	76.44

Feature representations are a crucial component of face clustering in videos. If the representation is robust, we can expect that the face tracks of each identity will be merged together in a unique cluster. Therefore, we recommend the use of self-supervised discriminative methods, TSiam and SSiam, over the unsupervised generative method, VAE.

In Figure 3.6 and Figure 3.7, we show the distribution of pairwise cosine similarities between tracks of same identity and different identity for the base features, TSiam and SSiam. We can observe that TSiam makes positive pairs have a very strong peak due to track-level supervision. At the same time SSiam (BF-0502) is hard to interpret using the histogram of similarity between tracks - this maybe due to absence of any supervision.

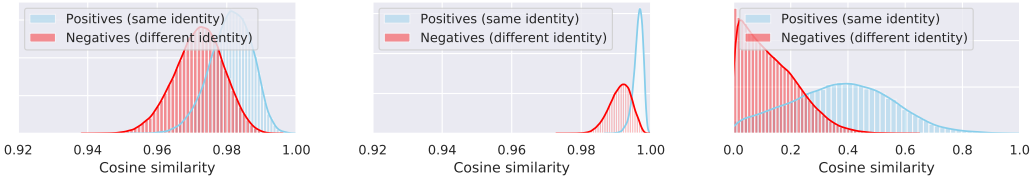
Do constraints help to merge tracks in face clustering? Conventional techniques for face clustering use hand-crafted features that are not very effective in the presence of illumination, and viewpoint variations. In this setting, *must-link* and *must-not-link* pairwise constraints are useful. However, when the feature representation is trained in a discriminative manner, one can obtain a similar or even better clustering performance without using these constraints. We hypothesize that any modeling performed on a powerful representation is complimentary to using such constraints, and hence leads to a better face grouping. We have shown, under such a setting, the face representation can be readily used with simple offline features and learning an efficient method to model additional constraints is meaningful.

An important consideration in clustering is to automatically infer the number of clusters along with the clustering. As numbers saturate, we hope the community moves towards this challenging problem (Tapaswi et al., 2019). We are working towards methods that can learn and infer an optimal number of clusters while



(a) BBT-0101 - Base (b) BBT-0101 - TSiam (c) BBT-0101 - SSiam

Figure 3.6: Histograms of pairwise cosine similarity between tracks of same identity (pos) and different identity (neg) for BBT-0101.



(a) BF-0502 - Base (b) BF-0502 - TSiam (c) BF-0502 - SSiam

Figure 3.7: Histograms of pairwise cosine similarity between tracks of same identity (pos) and different identity (neg) for BF-0502.

effectively learning representations. Our work is a hint towards achieving this goal without relying on explicit external constraints as the feature representations are discriminative enough to learn the data grouping with relaxed thresholds.

3.6 SUMMARY

In this chapter, we proposed self-supervised approaches for face clustering in videos, by distilling the identity factor from deep face representations. We showed that discriminative models can leverage dynamic generation of positive/negative constraints based on ordered face distances and do not have to only rely on track-level information that is typically used. We also presented Variational Autoencoder as a strong generative model that can learn the underlying distribution of face representations, and model identity as the latent variable. Our proposed models are unsupervised (or use automatically generated labels) and can be trained and evaluated efficiently as they involve only a few matrix multiplications.

We conducted experiments on three challenging video datasets. We observed that VAEs are able to generalize well when they have seen the set of characters (*e.g.* across episodes of a series), while discriminative models performed better when generalizing

to new series. Overall, our models are fast to train and evaluate and outperform the state-of-the-art while operating on datasets that contain more tracks with higher diversity in appearance. In the next chapter, we consider a way to generate weak labels from a clustering algorithm and show how they can be used efficiently to improve video face clustering

Chapter 4

Clustering-based Pair Generation

In the previous chapter, we opted for a method to mine pseudo-labels by sorting distances on a subset of face images. In this chapter, we consider a different approach to generating weak-labels using a clustering algorithm. A good clustering algorithm can discover natural groupings in data. These groupings, if used wisely, can provide weak supervision for learning representations. In this chapter, we present Clustering-based Contrastive Learning (*CCL*), a new clustering-based representation learning approach that uses labels obtained from clustering along with video constraints to learn discriminative face features. We demonstrate our method on the challenging task of learning representations for video face clustering. Through several ablation studies, we analyze the impact of creating pair-wise positive and negative labels from different sources. Experiments on three challenging video face clustering datasets, BBT-0101, BF-0502 and ACCIO, show that CCL achieves a new state-of-the-art on all datasets. An implementation is available at *https://github.com/ssarfraz/FINCH-Clustering*.

The content of this chapter is based on the following publication:

- Vivek Sharma, Makarand Tapaswi, M Saquib Sarfraz, and Rainer Stiefelhagen. **"Clustering based contrastive learning for improving face representations"**. In IEEE International Conference on Automatic Face and Gesture Recognition (FG), 2020.

4.1 INTRODUCTION

Learning strong and discriminative representations is important for diverse applications such as face analysis, medical imaging, and several other computer vision and natural language processing (NLP) tasks. While considerable progress has been made using deep neural networks, learning a representation often requires a large-scale dataset with manually curated ground-truth labels. To harness the power of deep networks on smaller datasets and tasks, pre-trained models (*e.g.* ResNet-101 (He et al., 2016) trained on ImageNet, VGG-Face (Parkhi et al., 2015) trained on a large number of face images) are often used as a feature extractors or fine-tuned for the new task.

We introduce a clustering-based learning method *CCL* to obtain a strong face representation on top of features extracted from a deep CNN. An ideal representation has small distances between samples from the same class, and large inter-class distances in feature space. Different from a fully-supervised setting where ground-truth labels are often provided by humans, we view CCL as a *self-supervised* learning paradigm where ground-truth labels are obtained automatically based on the structure of the dataset.

Self-supervised learning methods are receiving increasing attention as collecting large-scale datasets is extremely expensive. In NLP, word and sentence representations (*e.g.* word2vec (Mikolov et al., 2013), BERT (Devlin et al., 2018)) are often learned by modeling the structure of the sentence. In vision, there are efforts to learn image representations by leveraging spatial context (Doersch et al., 2015), ordering frames in a short video clip (Fernando et al., 2017; Misra et al., 2016), or tracking image regions (Wang and Gupta, 2015). There are also efforts to learn multimodal video representations by ordering video clips (Sharma et al., 2019b).

Among recent works in the *self-supervised* learning paradigm, (Caron et al., 2018; Guo et al., 2017; Jiang et al., 2017; Xie et al., 2016; Yang et al., 2016) propose to learn image representations (often an auto-encoder) using a clustering algorithm as objective, or as a means of providing pseudo-labels for training a deep model. One challenge with the above methods is that the models need to know the number of clusters (usually target number of clusters) a priori. Our chapter is related to the above as we utilize clustering algorithms to discover the structure of the data, however,

we do not assume knowledge of the number of clusters. In fact, our clustering setup yields a large number of clusters with high purity and few samples per cluster.

Our primary contribution, CCL, is a method for refining feature representations obtained using a deep CNN by discovering dataset clusters with high purity (typically few samples per cluster) and using these cluster labels as (potentially noisy) supervision. In particular, we obtain positive (same class) and negative (different class) pairs of samples using the cluster labels and train a Siamese network. We adopt the recently proposed FINCH clustering algorithm (Sarfraz et al., 2019) as a backbone since it provides hierarchical partitions with high purity, does not need any hyper-parameters, and has a low computational overhead while being extremely fast.

Towards our task of clustering faces in videos, we demonstrate how cluster labels can be integrated with video constraints to obtain positive (within and across neighboring clusters) and negative pairs (co-occurring faces and samples from farthest clusters). Both the cluster labels and video constraints are used to learn an effective face representation.

Our proposed approach is evaluated on three challenging benchmark video face clustering datasets: Big Bang Theory (BBT), Buffy the Vampire Slayer (BF) and Harry Potter (ACCIO). We empirically observe that deep features can be further refined by using weak cluster labels and video constraints leading to state-of-the-art performance. Importantly, we also discuss why the cluster labels may be more effective than using labels from alternative groupings such as tracking (*e.g.* TSiam (Sharma et al., 2019a)).

The remainder of this chapter is structured as follows: Section 4.2 provides an overview of related work. In Section 4.3, we propose our method for clustering-based representation learning, CCL. Extensive experiments, an ablation study, and comparison to the state-of-the-art are presented in Section 4.4. Finally, we conclude our chapter in Section 4.5.

4.2 Related Work

This section discusses face clustering and identification in videos. We primarily present methods for face representation learning with CNNs, followed by a short overview of FINCH-clustering algorithm and learning from clustering.

Video face clustering methods often follow 2 steps: obtain pairwise constraints typically by analyzing tracks, followed by representation/metric learning approaches and clustering. We present models through a historical perspective, with CNNs.

Face representation learning with CNNs. With the popularity of Convolutional Neural Networks (CNNs), there is a growing emphasis on improving face track representations using CNNs. An improved triplet loss that pushes the positive and negative samples apart in addition to the anchor relations is used by (Zhang et al., 2016a) to fine-tune a CNN. Zhang et al. (2016b) (JFAC) use a Markov Random Field to discover dynamic constraints iteratively to learn better representations during the clustering process. Inverse reinforcement learning on a ground-truth data is also used to learn a reward function that decides whether to merge a given pair of face features (He et al., 2018). In a slightly different vein, the problem of face detection and clustering is addressed jointly by (Jin et al., 2017), where a link-based clustering algorithm based on rank-1 counts verification merges frames based on a learned threshold.

Among recent works, Tapaswi et al. (2019) aim to estimate the number of clusters and their assignment and learn an embedding space that creates a fixed-radius balls for each character. Datta et al. (2018) use video constraints to generate a set of positive and negative face pairs. This chapter is related to the previous Chapter 3 but differs substantially in technical approach, where we proposed two approaches: *SSiam*, a distance matrix between features on a subset of frames is used to generate hard positive/negative face pairs; and *TSiam* uses temporal constraints and mines negative pairs for singleton tracks by exploiting track-level distances. Features are fine-tuned using a Siamese network with contrastive loss (Hadsell et al., 2006).

Different from related work, we propose a simple, yet effective approach where weak labels are generated using a clustering algorithm that is independent of video/track level constraints to learn discriminative face representations. In particular while He et al. (2018); Tapaswi et al. (2019) require ground-truth labels to learn embeddings, CCL does not.

FINCH clustering algorithm. Sarfraz et al. (2019) propose a new clustering algorithm (FINCH) based on first neighbor relations. FINCH does not require training and relies on good representations to discover a hierarchy of partitions. While FINCH demonstrates that pseudo labels can be used to train simple classifiers

with cross-entropy loss (Sarfraz et al., 2019), such a setup is challenging to use in the context of faces. When using a partition at high purity with many more clusters than classes, faces of one character belong to multiple clusters, and the CE loss pushes these clusters and their samples apart – an undesirable effect for our goal of clustering. Thus, in our work, we use FINCH to generate weak positive and negative face pairs that are used to improve video face clustering.

Clustering based learning. Recently clustering is also used to learn an embedding of data (Caron et al., 2018; Guo et al., 2017; Jiang et al., 2017; Sarfraz and Khan, 2011; Yang et al., 2016). Almost all of these methods cluster data samples at each forward pass (or at pre-determined epochs) into a given number of clusters. Subsequently, generated cluster labels or a loss over the clustering function are used to train a deep model (usually an auto-encoder). These methods require to specify the number of clusters, that is often the target number of classes. This is different from CCL that leverages a large number of small and pure clusters without specifying the particular number of clusters.

In summary, we show how pseudo-labels from clustering can be used effectively: (i) without needing to know the number of clusters, and (ii) by creating positive/negative pairs at a high-purity stage of clustering.

4.3 Clustering based Representation Learning

We present our method to improve the face representation automatically using weak cluster labels instead of manual annotations. We start this section by introducing the notation used throughout the remainder of the chapter. We then present a short overview of FINCH, followed by how we use it to obtain automatic labels. Finally, we discuss how we train our embedding and perform inference at test time. Algorithm 1 provides an overview of the steps in the proposed method.

4.3.1 Preliminaries

Let the dataset contain N face tracks $\mathcal{D} = \{(T_i, y_i)_{i=1}^{N}\}$, where each track T_i has a variable number K of face images, *i.e.* $T_i = \{(f_i^k)_{k=1}^{K}\}$. $y_i \in \{1, \ldots, C\}$ is the label

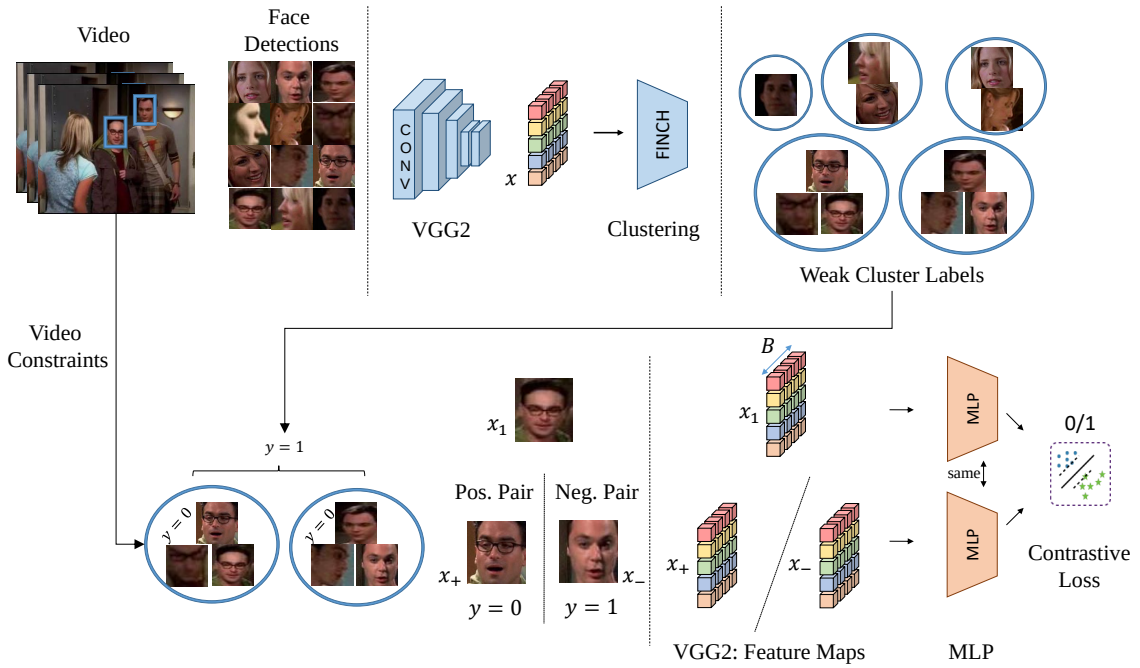


Figure 4.1: CCL training overview. Given a video with several face detections (top left), we first start by extracting features using a deep CNN and perform clustering using FINCH to obtain a large number of small but highly pure clusters (top). We create several positive and negative face image pairs using these cluster labels and train an MLP to further improve the feature representation using the contrative loss (bottom). At test time, the MLP is used as an embedding, and we cluster our samples using Hierarchical Agglomerative Clustering (HAC).

corresponding to the track T_i, where C is the total number of characters in the video. Please note that tracks are not necessary for the proposed learning approach, however, they are used during a part of the evaluation.

We use a CNN trained on face recognition (VGG2Face (Cao et al., 2018)) and extract features for each face image in the track from the penultimate layer (before classification) of the network. The features for a track are vectors $\{\mathbf{x}_i^1, \ldots, \mathbf{x}_i^K\}$ of size $\mathbf{x}_i^k \in \mathbb{R}^{D\times 1}$, where D denotes the feature dimension of the CNN feature maps. We refer to these as base features as they are not refined further by a learning approach.

Track-level representations are formed by an aggregation function $\Phi : \{\mathbf{x}_i^1, \ldots, \mathbf{x}_i^K\} \rightarrow \mathbf{v}_i$, that combines K frames to produce a single feature vector $\mathbf{v}_i \in \mathbb{R}^{D\times 1}$. While there are several methods of aggregation such as Fisher Vectors (Parkhi et al., 2014), Covariance Learning (Wang et al., 2012), Quality Aware Networks (Liu et al., 2017) or Neural Aggregation Networks (Yang et al., 2017a), Temporal 3D Convolution (Diba

et al., 2018a), we find that simple average pooling often works quite well. In our case, averaging allows to learn using features at the frame-level (without requiring tracking), but evaluate at the track-level.

We additionally ℓ_2 normalize the features to be unit vectors before using them for clustering, *i.e.* $\|\mathbf{v}_i\|_2 = 1$. Several previous works in this area (Sharma et al., 2017, 2019a; Tapaswi et al., 2019; Zhang et al., 2016a,b) have used Hierarchical Agglomerative Clustering (HAC) as the preferred technique for clustering. For a fair comparison, we too use HAC with the stopping condition set to the number of clusters equalling the number of main cast (C) in the video. We use the minimum variance ward linkage (Ward Jr., 1963) for all methods.

4.3.2 Clustering Algorithm

As part of our core method to obtain weak labels from clustering, we adopt the recently proposed FINCH algorithm (Sarfraz et al., 2019). FINCH belongs to the family of HAC algorithms and automatically discovers groupings in the data without requiring hyper-parameters or a priori knowledge such as the required number of clusters. The output of FINCH is a *small* set of partitions that provide a fine to coarse view on the discovered clustering. Note that this is different from classical HAC methods where each iteration merges one sample with existing clusters providing a complete list of partitions ranging from N clusters (all samples in their own cluster) to 1 cluster (all samples in one cluster).

FINCH works by linking first neighbors of each sample. Specifically, an adjacency link matrix is built between all pairs of samples as

$$A(i,j) = \begin{cases} 1 & \text{if } j = \kappa_i^1 \text{ or } \kappa_j^1 = i \text{ or } \kappa_i^1 = \kappa_j^1 \\ 0 & \text{otherwise} \end{cases} \tag{4.1}$$

where κ_i^1 represents the first neighbor of sample i. The adjacency matrix joins each sample i to their first neighbors via $j = \kappa_i^1$, enforces symmetry through $\kappa_j^1 = i$, and links samples that share a common first neighbor $\kappa_i^1 = \kappa_j^1$. Clustering is performed by identifying the connected components in the adjacency matrix.

Using FINCH to obtain weak labels. We use clusters from the second partition of the FINCH algorithm to mine positive and negative pairs. There are two differences from previous methods (Sarfraz et al., 2019) that have used clustering labels as a form of category labels: (i) we achieve a very high purity (usually over 99%) within each cluster; and (ii) the number of clusters is much greater than the true number of categories, and each cluster typically contains few samples (< 10).

The first partition corresponds to linking samples through the first neighbor relations, while the second partition links clusters created in the first step. Using the second partition provides increases diversity, without compromising on quality of the labels. We find that most samples clustered in the first partition are from within a track (*e.g.* neighboring frames), while those in the second partition are from across tracks (*cf.* Fig. 4.2). Empirically, we analyze the impact of choosing different partitions through ablation studies.

We treat each face image as a sample and perform clustering with FINCH to obtain the second partition of clusters $P_2 = \{G_1, \ldots, G_M\}$. Based on the cluster labels, we obtain pairwise positive and negative labels as follows:

- **PosC**: All face images in a cluster are considered for positive pairs. For clusters that have less than 10 samples, we also create positive pairs by combining samples from the current cluster G_{p1} with randomly sampled frames from another cluster G_{p2}, where G_{p2} is among the $Z = 25$ nearest clusters to G_{p1}.

- **NegC**: We create negative pairs by combining samples from a cluster G_{n1} with randomly sampled frames from a different cluster G_{n2}, where G_{n2} is among the Z farthest clusters to G_{n1}.

Impact of clustering approach. We use K-means as an alternative approach to obtain clusters that provide weak labels. Note that K-means has an important disadvantage wherein we need to know the number of clusters before hand. For a fair comparison, we use the number of clusters estimated by FINCH at the second partition. Specifically, we use *MiniBatch K-means* (Sculley, 2010) that is more suitable to work with large datasets. Our analysis shows that FINCH provides purer clusters that yield more accurate pairwise labels.

4.3.3 Video Level Constraints

Video face clustering often employs face tracking to link face detections made in a sequence of consecutive frames. Note that tracking can be thought of as a form of clustering in a small temporal window (typically within a shot). Using tracks, positive pairs are created by sampling frames within a track, while negative pairs are created by analyzing co-occurring tracks (Cinbis et al., 2011; Datta et al., 2018; Tapaswi et al., 2014b; Wu et al., 2013b).

In this chapter, we consider video constraints that can be derived at a frame-level (*i.e.* without tracking). We include co-occurring face images appearing in the same frame as potential negative pairs (**NVid**). About 50%-70% of the face images appear as singletons in a frame (as shown by TSiam (Sharma et al., 2019a)), and do not yield negative pairs through the video level constraints. Thus, being able to sample negative pairs from the singleton tracks using the cluster labels as discussed above is beneficial.

In addition to sampling negative pairs, we also use co-occurrence to correct clusters provided by FINCH. For example, if a cluster contains samples from a known negative pair, we keep the sample closest to the cluster mean, and create a new cluster using the rejected sample.

4.3.4 Training and Inference

We use the weak cluster labels and video constraints to obtain positive and negative face pairs for training a shallow Siamese network with Contrastive loss. We sample a pair of face representations $\mathbf{x}_1$ and $\mathbf{x}_2$ with $y = 0$ for a positive pair and $y = 1$ when corresponding to a negative pair. We train a multi-layer perceptron Q_θ using Contrastive Loss (Hadsell et al., 2006) as

$$\begin{aligned} &\mathcal{L}\left(W, y, Q_\theta(\mathbf{x}_1), Q_\theta(\mathbf{x}_2)\right) = \\ &\quad \frac{1}{2}\left((1-y)\cdot(d_W)^2 + y\cdot(\max(0, m - d_W))^2\right), \end{aligned} \tag{4.2}$$

where $W \in \mathbb{R}^{D\times d}$ is a linear layer that embeds $Q_\theta(\cdot)$ such that $d \ll D$ (in our case, $d = 2$). Here, each face image is encoded as $Q_\theta(\mathbf{x})$, where θ corresponds to the

Algorithm 1: Overview of CCL.

Input: Feature maps from VGG2Face CNN. $X = \{(\mathbf{x}_n)_{n=1}^{N}\} \in \mathbb{R}^D$. N is total number of faces. D is feature dimension.
Output: Final clustering of the N face samples.
Procedure:
1. Compute the partitions using FINCH clustering algorithm. $P_{1:L} = \text{FINCH}(X)$.
2. Select second partition $P_2 = \{G_1, \ldots, G_M\}$.
3. Train model parameters for 20 epochs:
– Sample positive and negative pairs for a batch.
$$(x_{p1}, x_{p2}) \sim (G_{p1}, G_{p2}) \text{ and } (x_{n1}, x_{n2}) \sim (G_{n1}, G_{n2}).$$
– Train MLP $Q_\theta(\cdot)$ using Contrastive loss.
4. Compute embeddings for each face image using $Q_\theta(\mathbf{x})$ and perform HAC to get C (# main cast) clusters.

trainable parameters. We find that $Q_\theta(\cdot)$ performs well when using a single linear layer. d_W is the Euclidean distance $d_W = \|W \cdot Q_\theta(\mathbf{x}_1) - W \cdot Q_\theta(\mathbf{x}_2)\|^2$, and m is the margin, empirically chosen to be 1. Fig. 4.1 illustrates this learning procedure.

During inference, we compute embeddings for face images using the MLP trained above. For frame-level evaluation, we cluster the features for each face image, while for track-level evaluation, we first aggregate image features using mean pool, followed by HAC. We report the clustering performance at the ground truth number of clusters (*i.e.* the number of main characters in the video). Algorithm 1 provides an overview of the entire training and inference procedure.

4.3.5 Implementation Details

CNN. We employ the VGG-2 face CNN (Cao et al., 2018) (ResNet-50) pre-trained on MS-Celeb-1M (Guo et al., 2016) and fine-tuned on 3.31 million face images of 9,131 subjects (VGG2 data). We extract `pool5_7x7_s1` features by first resizing RGB face images to 224×224 and pushing them through the CNN, resulting in $\mathbf{x}_k^i \in \mathbb{R}^{2048}$. At test time, we compute face embeddings using the learned MLP Q_θ and ℓ_2 normalize before HAC.

Siamese network MLP. For a fair comparison with Chapter 3, we use the same MLP network architecture. The network comprises of fully-connected layers: $\mathbb{R}^{2048} \rightarrow \mathbb{R}^{256} \rightarrow \mathbb{R}^2$. Note that the second linear layer is part of the contrastive

loss (corresponds to W in Eq. 4.2), and we use the feature representations at $\mathbb{R}^{256}$ for clustering same as Chapter 3. The model is trained using the Contrastive loss, and parameters are updated using Adam optimizer. We initialize the learning rate with 10^{-5} and decrease it by a factor of 10 every 15 epochs. We train the model for 20 epochs. We use Batch Normalization (BN).

Sampling positive and negative pairs for training. An epoch corresponds to sampling pairs from all clusters created by the FINCH algorithm. We consider 5 clusters per batch. For each data point in a cluster, we randomly sample from the same (or nearest) cluster and create one positive pair, and sample from the farthest clusters to create two negative pairs. Finally, we randomly subsample the above list to obtain 25 positive and 25 negative pairs for each cluster, resulting in a batch with 250 pairs (125 positive/negative).

4.4 Evaluation

We present our evaluation on three challenging datasets. We first describe the clustering metric, followed by a thorough analysis of the proposed method on, and end this section with a comparison of our approach against state-of-the-art.

Datasets and metric. We present our evaluation on three challenging datasets: BBT, BF and ACCIO, discussed in Section 2.3. We summarize the statistics of datasets in Table 2.1. In terms of evaluation metric following the previous works, for BBT and BF we report clustering accuracy (ACC or WCP), and for ACCIO in addition to ACC, we report BCubed Precision (P), Recall (R) and F-measure (F), as discussed in Section 2.4.

4.4.1 Clustering Performance and Generalization

Comparison against baselines. We present a thorough comparison of CCL against related baselines that employ a similar learning scheme.

In pseudo-relevance feedback (Yan et al., 2003a,b) (PRF), samples are treated independent of each other, and the pairs are created by considering closest positives

Table 4.1: Clustering accuracy of CCL and comparison to PRF (Yan et al., 2003b) TSiam (Sharma et al., 2019a) and SSiam (Sharma et al., 2019a) at track-level. Comparison against all evaluated models.

Train/Test	Base	PRF	TSiam	SSiam	CCL
BBT-0101	0.932	0.930	0.964	0.962	**0.982**
BF-0502	0.836	0.814	0.893	0.909	**0.921**

Table 4.2: Clustering accuracy of CCL and comparison to TSiam (Sharma et al., 2019a) and SSiam (Sharma et al., 2019a) at track-level, extending to all named characters. BBT-0101 has 5 main and 6 named characters; BF-0502 has 6 main and 12 named characters.

	BBT-0101			BF-0502		
	TSiam	SSiam	CCL	TSiam	SSiam	CCL
Main cast	0.964	0.962	**0.982**	0.893	0.909	**0.921**
All named	0.958	0.922	**0.966**	0.829	0.870	**0.903**

and farthest negatives. However, these pairs often provide negligible training signal (gradients) as they already conform to the loss requirements.

SSiam (Sharma et al., 2019a) is an improved version of PRF with batch processing where farthest positives and closest negatives are formed by looking at a batch of queries rather than whole dataset. This creates harder training samples that show improved performance over PRF.

In TSiam (Sharma et al., 2019a), positive pairs are formed by looking at samples within a track (a track is treated as a cluster of face images) and negative pairs by using co-occurrence and distances between track representations.

Finally, our proposed method CCL does not rely on tracking and uses pure clusters created by an automatic partitioning method (FINCH). We sort clusters by distances between their mean representations and then form positive and negative pairs.

Table 4.1 shows that CCL outperforms both SSiam and TSiam, and also provides significant gains over the base VGG2 features on BBT and BF. Fig. 4.2 (right) shows the number of faces in each cluster and the number of tracks that they are derived from. CCL's ability to obtain positive pairs from faces across multiple tracks (different from TSiam) might be an explanation for improved performance.

Table 4.3: A study on the impact of clustering algorithm. FINCH partitions are created for each dataset, and shown as separate table rows. In each row, results are presented for FINCH (above) and MiniBatch K-means (below). From left to right, Part. indicates the partition level of FINCH. #C is the total number of clusters in that partition as estimated by FINCH. Largest and smallest cluster sizes are indicated as LC/SC. Clustering purity of FINCH/K-means clusters (before CCL) is presented as ACC. L+/L- represents the number of samples correctly and wrongly clustered for the given partition. Finally, CCL-ACC is the performance of CCL: by training a model using weak labels from FINCH/K-means estimated clusters.

	BBT-0101					BF-0502					ACCIO				
Part.	#C	LC/SC	ACC	L+/L-	CCL-ACC	#C	LC/SC	ACC	L+/L-	CCL-ACC	#C	LC/SC	ACC	L+/L-	CCL-ACC
	41220				@#C=5	**39263**				@#C=6	**166885**				@#C=36
1	10156	48/2	0.997	41128/92	0.978	9677	42/2	0.994	39054/209	0.915	21444	3937/2	0.847	141496/25389	0.816
		1030/1	0.988	40744/476	0.971		758/1	0.986	38717/546	0.909		16041/1	0.899	150177/16708	0.841
2	**2236**	777/4	0.990	40809/411	**0.995**	**2167**	1346/4	0.978	38414/849	**0.938**	**3972**	33461/4	0.832	138903/27982	**0.837**
		1236/1	0.987	40687/533	0.991		1566/1	0.971	38161/1102	0.923		18076/1	0.866	144613/22272	**0.857**
3	490	2031/8	0.974	40149/1071	0.954	560	2574/9	0.957	37588/1675	0.922	944	41405/13	0.812	135543/31342	0.801
		1568/1	0.979	40367/853	0.957		2120/1	0.962	37806/1457	0.926		9838/1	0.845	141036/25849	0.813
4	101	7258/32	0.968	39936/1284	0.943	127	9335/27	0.923	36260/3003	0.903	161	70825/39	0.784	130987/35898	0.767
		2019/1	0.981	40475/745	0.945		2360/1	0.956	37555/1708	0.923		16502/1	0.800	133639/33246	0.782
5	13	10.9K/0.K	0.968	39.9K/1.2K	0.943	24	13.7K/0.2K	0.895	35.1K/4.0K	0.869	24	84.8K/0.2K	0.690	115.1K/51.7K	0.652
		7.3K/1	0.966	39.8K/1.3K	0.931		3.7K/0.2K	0.920	36.1K/3.1K	0.875		26.7K/1	0.724	120.9K/45.9K	0.696
6	2	26.4K/14.7K	0.724	29.8K/11.3K	-	5	20.8K/1.2K	0.799	31.3K/7.8K	-	4	152.4K/1.1K	0.387	64.6K/102.2K	-
		26.5K/14.7K	0.727	29.9K/11.2K	-		16.7K/3.3K	0.910	35.7K/3.5K	-		60.8K/17.2K	0.506	84.4K/82.4K	-
7	1	41.2K	0.372	15.3K/25.8K	-	1	39.3K	0.361	14.1K/25.0K	-	1	166.9K	0.309	51.6K/115.2K	-
		41.2K	0.372	15.3K/25.8K	-		39.3K	0.361	14.1K/25.0K	-		166.9K	0.309	51.6K/115.2K	-

Table 4.4: Impact of mining positive and negative pairs from different sources. Track-level accuracy of CCL.

		PosC	NegC	NVid	BBT-0101	BF-0502
1	Base	-	-	-	0.932	0.836
2		✓	-	-	0.951	0.859
3		-	✓	-	0.968	0.883
4		✓	-	✓	0.956	0.865
5		✓	✓	-	0.971	0.896
6		-	✓	✓	0.979	0.917
7	CCL	✓	✓	✓	**0.982**	**0.921**

Generalization to unseen characters. We further study how CCL can cope with adding unseen characters at test time by clustering all named characters in an episode. Results from Table 4.2 show that CCL is better poised at clustering new characters and achieves higher performance in all cases. We believe that CCL's performance scales well as it is trained on a diverse set of pairs and can generalize to unseen characters.

4.4.2 Sources of Positive and Negative Pairs

In Table 4.4, we present the importance of each source from which we obtain positive and negative pairs. Positive pairs obtained from clusters are denoted as **PosC**; negative pairs from clusters as **NegC**; and negative pairs using video constraints as **NVid**.

Rows 2 to 6 evaluate different combinations of the sources of pairs and show their relative importance. It can be seen that negative pairs alone (row 3) are more important than positive pairs (row 2). Additionally, using pairs from the clusters alone (row 5) provides performance similar to TSiam and SSiam (refer to Table 4.1), and including negatives from videos (row 7) provides best performance.

4.4.3 Impact of Clustering Algorithm

We now present a study on the impact of the clustering algorithm used for obtaining weak labels (see Table 4.3). We first obtain the hierarchy of partitions by processing

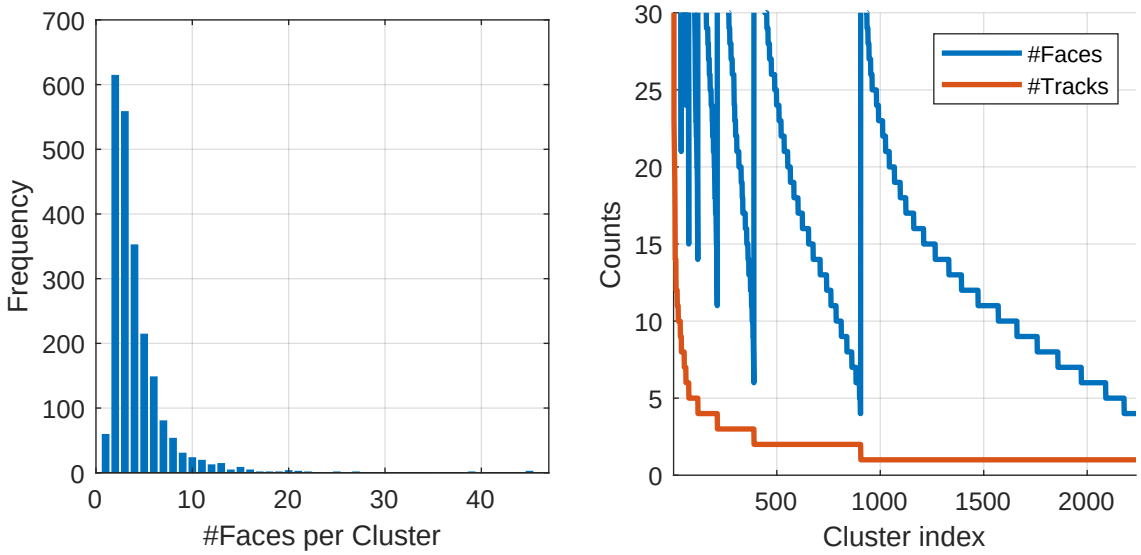


Figure 4.2: Key characteristics of FINCH (second partition) clustering for BBT-0101. *Left:* Histogram showing the number of faces in a cluster. Even if most clusters have less than 5 samples, we are able to obtain meaningful positive and negative pairs to train our model. *Right:* We plot the number of faces and number of tracks for each of the cluster indices (sorted by size for convenience). About 900 of the 2200 clusters created by FINCH contain faces from more than one track, leading to increased diversity of pairs.

each dataset with the FINCH algorithm. For a fair comparison, we use K-means with K set to the number of clusters provided by FINCH. Specifically, we use MiniBatch K-means for this evaluation as computing the full distance matrix would require $\sim$120 GB memory on larger datasets such as ACCIO.

We observe that FINCH not only automatically discovers meaningful partitions of the data but also achieves higher-performance (ACC in Table 4.3) as compared to K-means, especially at higher number of clusters. It is interesting to note the number of samples that are clustered correctly or wrongly (L+/L- in Table) as these play an important role while creating weak labels. Fig. 4.2 (left) shows that while the clusters are often very small (< 10 samples), they contain faces from more than one track (right). CCL with FINCH (second partition) obtains high performance after training with the labels (CCL-ACC) and outperforms labels provided by K-means clustering.

Running FINCH on larger datasets such as ACCIO takes $\sim$2 minutes (due to the use of fast nearest neighbors) as compared to MiniBatch K-means that takes $\sim$1 hour even though it is optimized for large datasets.

Table 4.5: Comparison to state-of-the-art with clustering accuracy (%) at frame level on both videos: BBT-0101 and BF-0502.

Method	BBT-0101	BF-0502
FINCH (CVPR '19) (Sarfraz et al., 2019)	99.16	92.73
ULDML (ICCV '11) (Cinbis et al., 2011)	57.00	41.62
HMRF (CVPR '13) (Wu et al., 2013b)	59.61	50.30
HMRF2 (ICCV '13) (Wu et al., 2013a)	66.77	–
WBSLRR (ECCV '14) (Xiao et al., 2014)	72.00	62.76
VDF (CVPR '17) (Sharma et al., 2017)	89.62	87.46
Imp-Triplet (PacRim '16) (Zhang et al., 2016a)	96.00	–
JFAC (ECCV '16) (Zhang et al., 2016b)	–	92.13
TSiam (FG '19) (Sharma et al., 2019a)	98.58	92.46
SSiam (FG '19) (Sharma et al., 2019a)	99.04	90.87
CCL (Ours with HAC)	**99.56**	**93.79**

4.4.4 COMPARISON WITH THE STATE-OF-THE-ART

Table 4.6: Frame-level clustering accuracy (%) of CCL and comparison to TSiam (Sharma et al., 2019a) and SSiam (Sharma et al., 2019a) over all datasets: BBT-0101, BF-0502 and ACCIO.

	#Clusters	Base	TSiam	SSiam	CCL
BBT-0101	5	94.00	98.58	99.04	**99.56**
BF-0502	6	91.20	92.46	90.87	**93.79**
ACCIO	36	79.90	81.30	82.00	**83.40**

While previous analysis has been done at the track-level, we now report frame-level clustering performance to present a fair comparison against previous work.

BBT and BF. We compare the results obtained with CCL to the current state-of-the-art approaches for video face clustering in Table 4.5. We report clustering accuracy (%) on two videos: BBT-0101 and BF-0502. We can observe that CCL outperforms previous approaches, achieving 0.52% and 1.33% absolute improvement in accuracy on BBT-0101 and BF-0502 respectively.

ACCIO. As common in previous works (Sharma et al., 2019a; Zhang et al., 2016b), we evaluate our method on the ACCIO dataset with 36 and 40 clusters for the 35

Table 4.7: Comparison of CCL with the state-of-the-art on ACCIO, evaluated at 36 and 40 clusters. Clustering accuracy at frame-level.

#Clusters = 36			
Methods	P	R	F
FINCH (CVPR '19) (Sarfraz et al., 2019)	0.748	0.677	0.711
JFAC (ECCV '16) (Zhang et al., 2016b)	0.690	0.350	0.460
TSiam (FG '19) (Sharma et al., 2019a)	0.749	0.382	0.506
SSiam (FG '19) (Sharma et al., 2019a)	0.766	0.386	0.514
CCL (Ours with HAC)	**0.779**	0.402	**0.530**
#Clusters = 40			
Methods	P	R	F
FINCH (CVPR '19) (Sarfraz et al., 2019)	0.733	0.711	0.722
DIFFRAC-DeepID2$^+$ (ICCV '11) (Zhang et al., 2016b)	0.557	0.213	0.301
WBSLRR-DeepID2$^+$ (ECCV '14) (Zhang et al., 2016b)	0.502	0.206	0.292
HMRF-DeepID2$^+$ (CVPR '13) (Zhang et al., 2016b)	0.599	0.230	0.332
JFAC (ECCV '16) (Zhang et al., 2016b)	0.711	0.352	0.471
TSiam (FG '19) (Sharma et al., 2019a)	0.763	0.362	0.491
SSiam (FG '19) (Sharma et al., 2019a)	0.777	0.371	0.502
CCL (Ours with HAC)	**0.786**	0.392	**0.523**

main characters – see Table 4.7. CCL significantly outperforms the state-of-the-art in unsupervised learning techniques and is comparable to Sarfraz et al. (2019). Note that FINCH (Sarfraz et al., 2019) is not trainable.

CCL vs. TSiam and SSiam. In Table 4.6, we report the frame-level clustering performance on all three datasets (as compared to track-level performance in Table 4.1). We observe that CCL outperforms TSiam, SSiam and also the base features by significant gains. CCL operates directly on face detections (like SSiam), while TSiam requires tracks.

Computational complexity. The time to compute FINCH partitions for BBT, BF, and ACCIO is approximately 45 seconds, and ∼2 minute respectively on CPU. Training CCL for 20 epochs on BBT-0101 requires less than half an hour on a GTX 1080 GPU using the PyTorch framework.

4.5 SUMMARY

In this chapter, we proposed a self-supervised, clustering-based contrastive learning approach for improving deep face representations. We showed that we can train discriminative models using positive and negative pairs obtained through clustering and video level constraints that do not rely on face tracking. Through experiments on three challenging datasets, we showed that CCL achieves state-of-the-art performance while being computationally efficient and easily scalable.

In the next chapter, we look at graph neural networks for self-supervised face grouping on graphs, wherein we utilize the video constraints to generate pseudo-labels.

Chapter 5

Self-Supervised Face-Grouping on Graphs

In the previous chapters, we opted for multilayer perceptron to learn discriminative features. In this chapter, we approach the problem using graph neural networks, the features communicate over the edges allowing each track's feature to exchange information with its neighbors.

Specifically, in this chapter, we propose a novel self-supervised method for fine-tuning deep face representations called Face-Grouping on Graphs. We apply our method to automatic face grouping, where characters are to be separated based on their identity. To solve this problem, a graph structure with positive and negative edges over a set of face-tracks based on their temporal overlap and similarity constraints is induced, which requires no manual labor. We compute feature representations over sub-sequences of each track (sub-tracks) in order to obtain robust features whilst being able to utilize information contained in face variance. Each sub-track is given the ability to exchange information with adjacent sub-tracks via a typed graph neural network running over the induced graph. This allows us to push each representation in a direction in feature space that groups all representations of the same character together and separates representations of different characters.

We show that our method is capable of improving clustering accuracy on popular video face clustering datasets *The Big Bang Theory* and *Buffy the Vampire Slayer* by 4.9% and 17.0% respectively compared to baseline performance, and 0.52%

respective 5.55% compared to state-of-the-art methods. Additionally, we achieve 19.0% absolute increase in B^3 F-Score on *Harry Potter 1 (ACCIO)* over other state-of-the-art unsupervised methods. We provide performance metrics on all episodes of *The Big Bang Theory* and *Buffy the Vampire Slayer* to enable further comparison in the future. An implementation is available at *https://github.com/RunOrVeith/FGG*.

The content of this chapter is based on the following publication:

- Veith Röthlingshöfer*, Vivek Sharma*, and Rainer Stiefelhagen. **"Self supervised face-grouping on graphs"**. In ACM International Conference on Multimedia, 2019. *Spotlight: oral presentation.*

V. Sharma came up with the main idea of learning graph like structure for face representation learning. V. Röthlingshöfer and V. Sharma contributed equally to the refining the whole idea, writing and experiments. V. Röthlingshöfer focused on the implementation.

5.1 INTRODUCTION

We address the problem of effectively learning and utilizing features to improve video face grouping. A good feature representation of positive face pairs should have small intra-person-distance, and large inter-person-distance compared to negative pairs in feature space. The feature space should be representative and permit better face clustering under unconstrained appearance variations (Cao et al., 2018; Sharma et al., 2019a; Tran et al., 2018; Wu et al., 2018). This implies that the solution of face grouping (or clustering) in videos will benefit potential applications in video understanding, video summarization, content-based indexing & retrieval and more. Due to the vast amount of video material created daily and the many different people that appear potentially, it is critical to use methods that do not require manual annotations and do not depend on the presence of specific actors.

Many previous works (Bäuml et al., 2013; Wu et al., 2013a; Zhang et al., 2016a,b) on unsupervised face grouping make use of pairwise constraints mined from the face-tracks, such as faces from the same track belonging to the same person, and faces from tracks overlapping in time belonging to different persons. Constraints based on similarity (Sharma et al., 2019a) or other sources (Tapaswi et al., 2012, 2014b) have also been considered. These constraints are then used as a pairwise

guide towards a suitable representation of each person in feature space using a loss function. The following scenario is a highly abstracted version of this process.

Imagine a room full of people (who represent the data points) with the goal to form groups within the room based on an attribute that each person was assigned. No one knows the attributes of anyone else except themselves. The strategy of previous approaches is to let an external observer pick a limited number of people, and tell each one the direction where to go based on constraints that the observer knows. The observer has not seen all attributes before starting. This requires many iterations to achieve a satisfactory grouping. If the people instead were allowed to communicate with others around them while knowing the pairwise constraints, they could form groups themselves in parallel without the need of an external observer. Information about people further away trickles through the room like a game of Chinese Whispers: "I talked to a group with a similar attribute as you, they went this way! My attribute is different, so I'm going to go the other way." is an example of information being passed through the room that allows everyone to find their correct group.

Motivated by recent works on graph neural networks, such as *message passing* (Gilmer et al., 2017) and more (Battaglia et al., 2018; Derr et al., 2018; Kipf and Welling, 2017), we propose Face-Grouping on Graphs (FGG), an approach to self-supervised face grouping that emulates the communication in the scenario above. The deep face features of each data point are laid out in a graph structure, where edges represent one of must-link/cannot-link constraints. Using a neural network that operates on the graph structure and differentiates between the different edge types, the features can exchange information and assimilate their position in feature space compared to other features belonging to the same person. Additionally, we propose to work with partially pooled features as a trade-off between robustness on track-level and conservation of variance information on frame-level. We call this representation subtrack-level; it is obtained by splitting each track into shorter segments. We make the following contributions:

1. We show that the exchange of information between different features via a graph structure provides valuable information to learn discriminative features.

2. The features obtained using FGG have high separability and outperform state-of-the-art unsupervised face grouping methods on all three examined benchmarks: A simple, well-lit dataset with few characters (*The Big Bang Theory* (Bäuml

et al., 2013; Tapaswi et al., 2012; Wu et al., 2013a; Zhang et al., 2016a)), a more difficult dataset containing many night-time shots (*Buffy - The Vampire Slayer* (Bäuml et al., 2013; Zhang et al., 2016b)), and *Harry Potter 1 (Accio)* (Ghaleb et al., 2015), a feature length film with many characters across many different lighting conditions.

3. We provide scores for previously unreported episodes of *The Big Bang Theory* and *Buffy - The Vampire Slayer* to enable further study in the future.

The remainder of this chapter is structured as follows: Section 5.2 gives an overview over related work. In Section 5.3, we discuss how a graph is constructed and how features can be learned. Extensive experiments, an ablation study and the comparison to the state-of-the-art are presented in Section 5.4 before concluding in Section 5.6.

5.2 Related Work

Learning on Graphs. Graphs are explored for a person-ID/retrieval problem in Huang et al. (2018). An input image of a person is given, and the goal is to find all instances of that person in a set of videos. They construct a graph over the input face-tracks using visual as well as temporal links and perform graph transduction (Subramanya and Bilmes, 2009; Wang et al., 2008) using the input image as labeled data. Note that their approach does neither update the feature representation, nor make use of graph convolution methods. Their construction of the graph over face-tracks is different to ours. Additionally, we try to learn features that separate identities when clustered, and they perform person search over a static feature set. Another work has used graph structures to encode the content of videos (Vicol et al., 2018), including person interactions, relationships, and many more attributes, such as reasons behind actions. In contrast, we only use the graph to structurize the information flow using interactions of characters appearing in a video.

Extending deep learning to work over non-euclidean structures has been a much-discussed topic recently (Battaglia et al., 2018; Bresson and Laurent, 2017; Bronstein et al., 2017; Derr et al., 2018; Gilmer et al., 2017; Kipf and Welling, 2017; Li et al., 2016). In Kipf and Welling (2017), the authors introduce a feasible approximation

scheme for spectral graph convolutions and introduce the Graph Convolutional Network (GCN). More generalizations, such as the *message passing* framework (Gilmer et al., 2017) and the more recent *graph network* framework (Battaglia et al., 2018) have been proposed. In the context of Battaglia et al. (2018), we use a *node-focused* approach, since we collect output features from each node in the graph. We base our network on Kipf and Welling (2017) with the modification of residuality from (Bresson and Laurent, 2017; He et al., 2016) and multiple edge types (Li et al., 2016).

Other works (Hamilton et al., 2017; Narayanan et al., 2017) are looking to find representations for graphs, i.e. encode the structure of a graph. We use graph structure to learn representations of face-tracks. In Cai et al. (2011), a nearest-neighbor graph structure is used to cluster documents in a matrix-factorization framework. Utilizing must-link and cannot-link constraints implies a *signed graph* with positive and negative edges. The authors of Derr et al. (2018) explore link prediction on signed graphs using balance theory: "The friend of my friend is my friend" and "The enemy of my friend is my enemy". We have direct edges satisfying this assumption, and thus do not have to learn it. We want to acknowledge the concurrent work of Wang et al. (2019), who tackle the problem of face grouping as link prediction in a graph using GCNs.

5.3 Method

5.3.1 Preliminaries

Assume a given set of face-tracks T obtained from a video sequence. Each track $t_{i_{a,b}} \in T, i \in [1, \ldots, |T|]$ consists of $b - a + 1$ individual, temporally ordered faces. The track starts at frame a and ends at frame b (inclusive). Each track consists of per-frame faces $t_{i_{[a,b],j}}, j \in [0, b - a + 1)$, that belong to a person c out of the set of possible persons C. Each face is represented by a deep feature $\boldsymbol{f}_{ij} \in \mathbb{R}^d, d \in \mathbb{N}^+$. We use features $\boldsymbol{f}_{ij} \in \mathbb{R}^{2048}$ from a VGG2 pretrained CNN (Cao et al., 2018).

In summary, $t_{i_{[a,b],j}} = \boldsymbol{f}_{ij}$ corresponds to the j^{th} face in the i^{th} track, which starts at frame a and ends at frame b. Note that a single face that has no temporal information is a special case where $a = b$, but can otherwise be treated the same as a face-track with temporal information. For notational convenience and readability, we drop (some of) the indices in the following where they are not needed.

We want to group all tracks that belong to the same person together into clusters $\hat{C}_1, \ldots, \hat{C}_{|C|}$, so that any $\hat{C}$ contains exactly all tracks belonging to a single person. The number of total clusters $|C|$ is assumed to be known in advance and is equal to the number of characters. A Graph Convolutional Network (GCN) (Kipf and Welling, 2017) (discussed in more detail in Section 5.3.3) is used to fine-tune the features, and *Hierarchical Agglomerative Clustering* (HAC) (Ward Jr., 1963) is used for grouping.

Next, we describe how a graph on which the GCN is trained can be obtained from the face-tracks.

5.3.2 Graph Construction

An undirected Graph $G = (V, E)$ can be constructed over T by representing each track t_i with a node and connecting the nodes using one of several constraints. We use t_i both for the track and the node that it represents. A node is associated with a mean-pooled deep feature $f_i \in \mathbb{R}^{2048}$ over a track:

$$\boldsymbol{f}_i = \text{mean-pool}(t_{i_{a,b}}) = \frac{1}{b-a+1} \sum_{j=0}^{b-a} \boldsymbol{f}_{ij} \tag{5.1}$$

We define two edge types with which any two nodes can be connected: We call the first kind *must-link* edge, and the second kind *cannot-link* edge in accordance with previous literature (Cour et al., 2010; Sharma et al., 2019a; Tapaswi et al., 2014b; Wu et al., 2013b). The different edge types are used by the GCN to differentiate whether features should either be brought closer together or pushed further apart. This allows information to flow differently across must-link and cannot-link edges, thus enabling each track to update its representation depending on the representation of its neighbors. We use separate learnable weight tensors per edge type and layer to enable this behavior (Li et al., 2016). Note that even though the edge names contain the word "must" and "cannot", they are not enforced strictly but serve as a soft guide towards a representation that fulfills these constraints. Each edge $(t_i, t_j, w_{ij}, k) \in E$ has an associated weight $w_{ij} \in \mathbb{R}$ that acts as a proxy for the certainty with which the edge is present, and an edge type $k \in \{must-link, cannot-link\}$. The initial graph contains no edges and all tracks: $V = T$ and $E = \varnothing$. Edges are created based on the following constraints. The constraints are used similarly to previous

works (Sharma et al., 2019a; Zhang et al., 2016a) to generate labels for the loss function (see Section 5.3.3). To the best of our knowledge, we are the first to additionally use these constraints to guide information flow between sub-tracks.

Temporal Constraints. Some information is inherent to the face-tracks: Under the assumption that no person appears more than once at a given time, it is guaranteed that face-tracks overlapping in time correspond to different persons. Thus, we add temporal cannot-link edges E_{tc} between any track pair that overlaps in time. Because these edges are known to be correct, their weight is always set to 1.

$$E_{tc} = \{(t_{i_{a,b}}, t_{j_{a',b'}}, 1, cannot-link) | t_{i_{a,b}} \in T, t_{j_{a',b'}} \in T, i \neq j, \\ [a,b] \cap [a',b'] \neq \varnothing\} \tag{5.2}$$

Must-link constraints are mined from the fact that each face-track contains only one person. In this formulation however, we can not add any temporal must-link edges E_{tm}, since each track corresponds to exactly one node. There are no two nodes for which it is known solely based on temporal constraints that they must correspond to the same person. To alleviate this issue, we propose an intermediate representation of the face-tracks: Each track $t_{i_{a,b}}$ is split into $s_i \in [1, \dots, b-a+1]$ disjunct, roughly evenly-sized sub-tracks $t_{i_{c_1,d_1}}, \dots, t_{i_{c_{s_i},d_{s_i}}}$ so that $\bigcup_{k=1}^{s_i} [c_k, d_k] = [a,b]$ and $\bigcap_{k=1}^{s_i} [c_k, d_k] = \varnothing$.

Applying the split operation

$$S(t_i, s_i) = \bigcup_{k=1}^{s_i} t_{i_{c_k,d_k}} \tag{5.3}$$

retrieves the set of resulting sub-tracks. Because any existing edges connected to t_i also apply to each sub-track, these edges are copied to each sub-track as E_{copy}:

$$e_{copy}(t_i, s_i) = S(t_i, s_i) \times \bigcup_{(t_i,v,w,k) \in E} (v, w, k) \tag{5.4}$$

$$E_{copy} = \bigcup_{i=0}^{|T|} e_{copy}(t_i, s_i) \tag{5.5}$$

Here, $\times$ indicates Cartesian product. A split graph is iteratively constructed by replacing the i^{th} node with all created sub-tracks $S(t_i, s_i)$ and updating the edges with the copied edges. Each split grows the graph by $s_i - 1$ nodes. This procedure

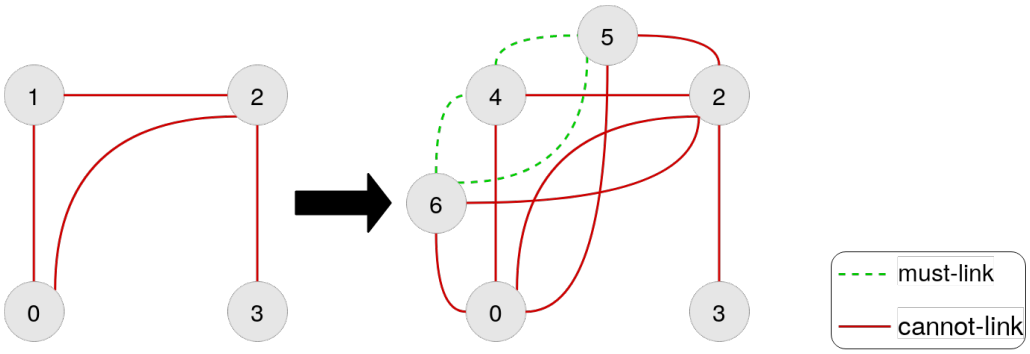


Figure 5.1: Exemplary split of a toy graph: On the left a graph $G = (S(T, \mathbf{1}), E_{tc})$ representing four temporally overlapping tracks. Each node represents a full track $t \in T$. No must-link edges are present. On the right $G' = (S(T, [1\ 3\ 1\ 1]), E_{tc} \cup E_{copy} \cup E_{tm})$, with track t_1 split into 3 sub-tracks. The created nodes are t_4, t_5 and t_6. All created nodes are completely connected with must-link edges, and they adopt the cannot-link edges of t_1.

is fully characterized by a split-vector $\mathbf{s} \in \bigtimes_{t_{a,b} \in T} [1, b - a + 1]$. We call this the *split construction* $S(T, \mathbf{s})$. Note that using $\mathbf{s} = \mathbf{1}$ is equivalent to the unsplit track-level graph, and using $\mathbf{s}^{\text{frame-level}} := \mathbf{s}_i = b - a + 1\ \forall t_{i_{a,b}} \in T$ corresponds to a graph on frame-level. When a track is split, the feature representation of each created node is the mean-pooled feature over the corresponding sub-track.

Working on a split graph allows to add temporal must-link edges E_{tm}. All sub-tracks of a given track t_i are connected with a must-link edge of weight 1:

$$e_{tm}(t_i, s_i) = (S(t_i, s_i) \times S(t_i, s_i)) \setminus \bigcup_{k=1}^{s_i} (t_{i_{c_k,d_k}}, t_{i_{c_k,d_k}}) \tag{5.6}$$

$$E_{tm} = \bigcup_{i=1}^{|T|} \bigcup_{t_j, t_k \in e_{tm}(t_i, s_i)} (t_j, t_k, 1, must - link) \tag{5.7}$$

The graph is now defined by $G = (S(T, \mathbf{s}), E_{tc} \cup E_{copy} \cup E_{tm})$. See Figure 5.1 for an example.

Split Variants. To find a suitable split $\mathbf{s}$, we must discuss the trade-offs that splitting has. The more nodes and edges the graph has, the bigger the feature matrix and adjacency matrix. This results in a higher run time and memory requirement, even when utilizing sparse matrix multiplications. Furthermore, a smaller graph allows information to flow across the graph faster. Since we utilize GCN (see Section 5.3.3), the receptive field for k layers is the k^{th} order neighborhood for each node (Kipf and Welling, 2017). This means that for information to propagate as far in a

split graph as in an un-split graph, more layers are required, which again increases the computational burden. When operating on a track-level graph, all must-link constraints are met automatically, because there is only one node to cluster per track. In contrast, it is possible for each sub-track to end up in a different cluster when splitting. We argue that this is in fact a positive effect. Mean representations provide protection against outliers on the one hand side. Splitting can be regarded as de-blurring on the other hand side: The mean feature over a full (long) face-track can not express the variance across multiple head poses and lighting conditions - resulting in a relatively uniform feature vector - whereas the same track split into a reasonable amount of sub-tracks retains information about the variance. More distinct features per person on the sub-track level make it more probable to find suitable clusters, instead of having to cluster similar features on track-level.

It is thus important to choose $\mathbf{s}$ in such a way to keep the run time low and minimize the influence of outliers, but split enough times to be able to learn from the intra-person variance within the features. We find a simple length-based heuristic $\mathbf{s}^h$ to work well across datasets. Given a number of maximal sub-tracks $X \in \mathbb{N}^+$, a step size $\Delta \in \mathbb{N}^+$ and a lower bound $B \in \mathbb{N}^+$ below which no split is applied, we define

$$s^h(t_{a,b}) = max(1, min(X, \frac{1 + (b - a + 1) - B}{\Delta})) \tag{5.8}$$

and $\mathbf{s}_i^h = s^h(t_i)$. Intuitively, this does not split a track if it is less than B frames long and splits into one more sub-track every additional Δ frames up to a maximum number of sub-tracks X.

Similarity Constraints. In addition to the temporal constraints that are inherent in the data and provide a learning signal for all co-occurring face-tracks, we explore mining additional edges for tracks that occur by themselves based on the cosine similarity of the input features. The approach is similar to Sharma et al. (2019a). Given a graph G, we sample a fraction $\alpha \in [0, 1]$ of the graph's isolated nodes and compute the pairwise cosine similarity between the sampled nodes' features, resulting in a similarity matrix $D \in [0, 1]^{n \times n}$, where n is the number of sampled nodes. A node is only an isolate node, if the corresponding track is too short to be split (less than B frames). With $\beta \in [0, 1]$ we then connect the $\frac{\beta}{2}$ most similar pairs with a must-link edge to obtain E_+ with the weight corresponding to the similarity score. The $\frac{\beta}{2}$ least similar pairs are connected with a cannot-link edge in E_-. The weight is set to

$1 - similarity$, since a low similarity indicates a high certainty for the cannot-link edge. Let $k = \frac{n\beta}{2}$.

$$E_+ = arg_k \min_{D_{ij}, i \neq j} (t_i, t_j, D_{ij}, must - link) \tag{5.9}$$

$$E_- = arg_k \max_{D_{ij}, i \neq j} (t_i, t_j, 1 - D_{ij}, cannot - link) \tag{5.10}$$

The final graph is obtained as

$$G = (S(T, \mathbf{s}), E_{tc} \cup E_{copy} \cup E_{tm} \cup E_+ \cup E_-) \tag{5.11}$$

5.3.3 TRAINING PROCEDURE

The procedure to group a set of face-tracks using our proposed Face-Grouping on Graphs (FGG) method is as follows: First, a graph is constructed over the face-tracks of a given episode. Each node represents a full track. Then, the cannot-link edges based on temporal overlap are added. Next, the temporal must-link edges are added by splitting the graph with heuristic $\mathbf{s}^h$. We fix $X = 10$ to keep the computation time short and find $B = 50, \Delta = 10$ to work well in practice. Finally, the remaining isolated tracks are connected with similarity-based edges. We use $\alpha = 1$ and $\beta = 0.03$, which was verified on a validation set to produce on average $\geq 99\%$ correct edges. A study on the influence of wrong edges is presented in Section 5.4.2.

The graph is then run through a graph neural network, which outputs a final feature embedding. Our model consists of the following layers: Fully-Connected (FC), a fully connected layer that only receives the features, but not the graph structure as input. It allows us to decrease computational cost of the following graph convolutions. *resGC* is a residual graph convolution layer (Bresson and Laurent, 2017; He et al., 2016; Kipf and Welling, 2017). Each node updates its representation based on the features in the neighboring nodes. It is important to differentiate whether a neighbor is connected with a must-link or a cannot-link edge, and how each edge type should influence the feature update. Thus, the graph convolution layers have a separate weight tensor for each edge type. The contributions of the different types are summed to obtain the combined output as suggested in (Li et al., 2016). The residual Graph Convolution (resGC) layers are initialized with the uniform He-method (He et al., 2015), which improves stability. *BN* is a batch norm layer with tracking of

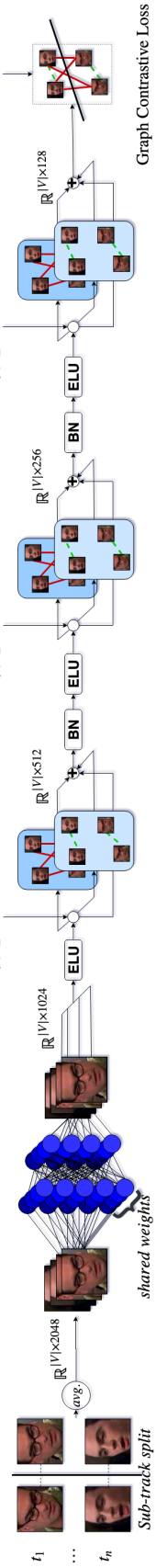


Figure 5.2: Our FGG model. The input is a matrix of all pooled sub-track features, the output is the updated representation of each sub-track. Note that the model never actually sees any images, they are just used to represent a sub-track here. The fully-connected layer uses the same weights for each track, i.e. the batch dimension is equivalent to $|V|$.

Table 5.1: The layer structure for a graph $G = (V, E)$. Note that $|V|$ is dynamic and changes depending on the input episode. With several design choices, we optimized efficiently this network architecture that generalized over all datasets.

Layer	Input Dim.		Output Dim.
	Features	Adjacency	
FC + ELU	$\|V\| \times 2048$	—	$\|V\| \times 1024$
resGC + BN + ELU	$\|V\| \times 1024$	$2 \times \|V\| \times \|V\|$	$\|V\| \times 512$
resGC + BN + ELU	$\|V\| \times 512$	$2 \times \|V\| \times \|V\|$	$\|V\| \times 256$
resGC	$\|V\| \times 256$	$2 \times \|V\| \times \|V\|$	$\|V\| \times 128$

running statistics disabled (Ioffe and Szegedy, 2015). We use a batch size of 1, which is the full graph of one episode. *ELU* is the Exponential Linear Unit activation function (Clevert et al., 2016). The model is described in detail in Table 5.1. See Figure 5.2 for a graphical description. In order to train the neural network, we use a Contrastive Loss (Hadsell et al., 2006) over connected nodes in the graph:

$$\begin{aligned} &\mathcal{L}(f_W, E, A^+, A^-, m) = \\ &\frac{1}{2|E|} \sum_{t_i, t_j \in E} (1[A^+_{ij} \neq 0] d(f_W(t_i), f_W(t_j))^2 + \\ &\qquad 1[A^-_{ij} \neq 0] max(0, m - d(f_W(t_i), f_W(t_j))^2)) \end{aligned} \tag{5.12}$$

Here, $f_W : \mathbb{R}^N \to \mathbb{R}^n$ embeds input features $\in \mathbb{R}^N$ into a lower dimensional feature space $\mathbb{R}^n$ using the trainable weights W. In our context, f_W is the neural network described in Table 5.1. d is the euclidean distance function. E is the set of edges in the graph, m is the margin parameter empirically set to 1, and A^+, A^- are the must-link and cannot-link adjacency matrices. 1 is the indicator function.

We use Adam (Kingma and Ba, 2015) as an optimizer and train for 30 epochs with a learning rate of 0.0001. One epoch consists of one step over each selected episode.

During testing, the resulting features are l_2-normalized and clustered with Hierarchical Agglomerative Clustering (Ward Jr., 1963) using the number of characters as the number of clusters. This procedure is highly extensible: Other clustering algorithms can be swapped in, and any possible additional (pairwise) constraints can be encoded by adding new edges, or even new edge types.

5.4 Experiments

In this section, we first introduce the datasets and implementation details of our proposed approach, followed by a thorough analysis of the proposed methods, and end this section with a comparison of our approach against state-of-the-art.

Datasets and metric. We present our evaluation on three challenging datasets: BBT, BF and ACCIO, discussed in Section 2.3. We summarize the statistics of datasets in Table 2.1. In terms of evaluation metric following the previous works, for BBT and BF we report clustering accuracy (ACC or WCP), and for ACCIO in addition to ACC, we report BCubed Precision (P), Recall (R) and F-measure (F), as discussed in Section 2.4.

5.4.1 Clustering Performance and Generalization

Base Features. In Table 5.2 we show WCP of each episode when clustering the VGG2 (Cao et al., 2018) features directly using HAC (Ward Jr., 1963). We exclude any non-main characters for fair comparison. Track-level results use mean-pooled frame-level features. The number of clusters is set to the number of main characters. Note that there is a significant spread in performance across episodes.

Performance on Training Videos. We report the clustering performance when training and testing on each episode in Table 5.2. Note that training is self-supervised through the temporal and similarity edges, and no ground-truth labels are required. Section 5.5 gives an empiric example of the high degree of separation achieved by our method. The training time for one episode is limited by the evaluation step, which requires clustering to be run. Training by itself takes 10 seconds per epoch on a GTX TitanX GPU plus a one-time cost of about 2 minutes to load the features into memory and construct the graph for a total of 7 minutes. The time per epoch could be improved by utilizing multi-dimensional sparse tensor multiplications, which are unavailable in PyTorch (version 1.0.1) (Paszke et al., 2017).

Intra- and Inter-Series Generalization. We evaluate our model on unseen episodes of the series that it has been trained on (intra-series generalization), and on episodes of a series that it has never seen before (inter-series generalization). In particular, this changes the number of characters. Table 5.3 shows the results of

Table 5.2: Comparison of WCP on each episode. The first row shows results when clustering the VGG2 (Cao et al., 2018) face representations of the main characters as they are contained in each dataset. The second row shows the performance of our proposed FGG model, averaged over 5 runs. Note that the standard deviation (std) is in units of permille (‰). The drop in performance for BBT-0106 is caused by the nature of that episode: The characters are at a Halloween party wearing costumes.

		BBT						**BF**					
		0101	0102	0103	0104	0105	0106	0501	0502	0503	0504	0505	0506
VGG2 base	track-level	0.947	0.984	0.996	0.875	0.936	0.888	0.891	0.812	0.954	0.905	0.866	0.804
	frame-level	0.936	0.968	0.924	0.981	0.920	0.925	0.927	0.903	0.947	0.887	0.905	0.843
FGG (5 runs)	mean	0.996	0.996	0.997	0.947	0.996	0.931	0.955	0.980	0.982	0.965	0.952	0.97
	std (in ‰)	1.2	0.9	0.7	31.3	1.5	1.3	14.5	6.3	4.1	16.5	14.5	6.0

both intra- and inter-series generalization between BBT and BF. Notably, using a model trained on an episode A can improve the performance on another episode B compared to a model that has been trained on B, if it achieves a high performance on A. For example, we found that a model trained on BBT-0101 achieves a WCP of 97.4% and 96.2% on BBT-0104 and BBT-0106 respectively —an increase of $\sim 3\%$ compared to the model trained *and* tested on BBT-0104 and BBT-0106 with 94.7% and 93.1% respectively (see Table 5.2). This indicates generalization ability within series. The inter-series generalization is significantly worse compared to previous approaches. Since the graph structure is a proxy for how often characters interact, which is different per series, this is expected. The network learns features that fit a given graph structure. Interactions within the same series are more likely to be similar across episodes, but can be completely different between different series. Note that our main goal is not generalization, and we do not use any measures to increase generalization ability such as dropout or introduce stochasticity in other ways.

Generalization to Unseen Characters. A further study on how FGG is able to cope with adding or removing characters is presented in Table 5.4. We cross-evaluate models trained on the main cast (5 for BBT-0101, 6 for BF-0502) and all named characters (6 for BBT-0101, 12 for BF-0502). On BBT-0101, the graph structure is highly similar, since only one character is added. This allows the model to obtain a high score. On BF-0502, the graph structure changes more dramatically, since nodes for tracks of six characters are added. This explains the greater drop in WCP compared to BBT-0101. However, much of the graph structure is still the same, allowing us to outperform previous methods (see Table 5.5). Note that the FGG model trained on the main cast and evaluated on all named characters achieves a higher WCP than Sharma et al. (2019a), even when compared to their evaluation on the main cast. Additionally, FGG can handle a reduction of characters from training on all named cast to evaluation on the main cast. Again, we can observe that in Table 5.5, BF-0502 is affected more since the graph structure changes more dramatically.

5.4.2 Ablation Study

In order to investigate how the graph structure is used by the model, we perform several ablation studies. First, edge dropout is used to determine how the model can

Table 5.3: WCP when evaluating across different episodes of the same dataset and a different dataset. We show values including and excluding the episode seen during training. Furthermore, we differentiate between evaluating each episode separately and then taking the mean (column *separate*), and clustering all features jointly (column *joint*). The best intra- and inter-series generalization is marked in bold.

Trained On	Method	BBT01[01-06]		BF05[01-06]		BBT01[02-06]		BF05[01,03-06]	
		separate	joint	separate	joint	separate	joint	separate	joint
BBT-0101	TSiam (Sharma et al., 2019a)	—	0.936	—	**0.875**	—	0.930	—	—
	SSiam (Sharma et al., 2019a)	—	0.922	—	0.862	—	0.914	—	—
	FGG	**0.983**	0.980	0.684	0.578	**0.980**	0.975	—	—
BF-0502	TSiam (Sharma et al., 2019a)	—	**0.915**	—	0.890	—	—	—	0.889
	SSiam (Sharma et al., 2019a)	—	0.883	—	0.905	—	—	—	0.904
	FGG	0.767	0.702	**0.933**	0.875	—	—	0.922	**0.932**

Table 5.4: Performance when evaluation on more and fewer characters than FGG is trained on. A comparison the previous works is presented in Table 5.5.

Train \ Test	BBT-0101		BF-0502	
	All named	Main	All named	Main
All named cast	0.994	0.997	0.931	0.943
Main cast	0.993	0.996	0.928	0.980

Table 5.5: Generalization to additional characters. We use the best model trained on the main cast as reported in Table 5.7. TSiam/SSiam and CCL are described in Chapter 3 (Sharma et al., 2019a) and Chapter 4 (Sharma et al., 2020) respectively.

	BBT-0101				BF-0502			
	TSiam	SSiam	CCL	FGG	TSiam	SSiam	CCL	FGG
Main cast	0.964	0.962	0.982	**0.996**	0.893	0.909	0.921	**0.980**
All named cast	0.958	0.922	0.966	**0.993**	0.829	0.870	0.903	**0.928**

cope with fewer edges. A dropout rate of $x\%$ is independent between must-link edges and of cannot-link edges, i.e. $x\%$ of must-link edges and $x\%$ of cannot-link edges are dropped out. We evaluate in 10% steps, and in 1% steps for dropout rates $\geq 90\%$.

Second, an experiment is conducted with regard to handling the addition of wrong edges. We add an increasing amount of wrong edges between previously unconnected nodes. A wrong edge is a must-link edge if the samples nodes belong to different persons, and a cannot-link edge otherwise. The weight for each additional edge is set to 1. We randomly sample $x\%$ of nodes and add wrong edges between all pairs in the sample where there is not already an existing edge. We stop after a memory error occurs due to the amount of edges. Both experiments can be seen in Figure 5.3. The model is insensitive to the amount of edges; stability decreases as edges are removed, but the overall performance trend only drops considerably after using a dropout probability of over 90%. At a dropout rate of 1, the model layers are not updated during training, since the graph contrastive loss (Eq. 5.12) is 0 when there are no edges present. The model then acts as a randomly initialized non-linear projection of the VGG2 features, which explains why the performance is not decreasing more. In contrast, it is very sensitive to the addition of wrong edges due to the spread of information over these edges.

Furthermore, different changes to the graph and model structure are evaluated: *FGG-track-level* uses the graph before being split into sub-tracks, and without any

similarity-based features. Thus, only cannot-link edges are present and the graph is on track-level. *FGG-track-level-similarity* is like *FGG-track-level*, but similarity-based edges are added. This improves the performance as well as the stability, especially on BF-0502, where there is a lower amount of cannot-link edges present (equal to the number of overlaps in Table 2.1). *FGG-random-split* uses all edge types, but each track is split into uniformly sampled random number of sub-tracks between 1 and 10 (or the maximum possible split for short tracks). In *FGG-cannot-link*, all must-link edges are removed; only cannot-link edges are present. *FGG-frame-level* splits each node as much as possible, i.e. uses a frame-level graph with all edge types. *FGG* is our full model using temporal must-link and cannot-link edges as well as similarity-based edges on the sub-track level graph as defined in Eq. 5.11. We use *FGG* as default model, as it performs the best.

Table 5.6 summarizes these experiments. The performance on BBT-0101 fluctuates considerably less than on BF-0502, indicating that it is a simpler dataset and that the base VGG2 features are already much more discriminative (as can also been seen in Table 5.2).

We analyse the impact of pooling over sub-tracks. To allow a meaningful comparison, the same split graph structure is used, but each node is associated with the mean-pooled feature of the *full* track. We compare the performance on all episodes of BBT and BF. On average, pooling over sub-tracks increases the WCP by 1.65%, confirming that information contained in the variance of features is beneficial to the overall performance.

Finally, we observe that pooling the learned subtrack-level features over the complete original track before clustering does not alter the performance. This indicates that passing information along the edges of the sub-tracks produces robust features.

5.4.3 COMPARISON TO STATE-OF-THE-ART

BBT and BF. We compare the results obtained with FGG to the current state-of-the-art approaches for video face clustering in Table 5.7. Other approaches are evaluated at frame-level, where we report track-level performance. Since splitting the tracks into shorter sub-tracks is a main component of our approach, we evaluate on sub-track level. We can observe that FGG outperforms previous approaches,

Table 5.6: WCP when training the FGG model with different features dis/enabled. Mean and standard deviation are computed across 5 runs. See Section 5.4.2 for an explanation of each experiment variation. † indicates OUT_OF_MEMORY.

Version	BBT-0101		BF-0502	
	mean	std	mean	std
FGG-track-level	98.48	0.99	83.73	1.00
FGG-track-level-similarity	99.28	0.18	93.80	0.64
FGG-random-split	98.24	0.58	92.01	0.77
FGG-cannot-link	99.36	0.23	85.43	4.67
FGG-must-link	99.16	0.11	97.42	0.71
FGG-frame-level	†	†	†	†
FGG	**99.56**	**0.12**	**98.01**	**0.63**

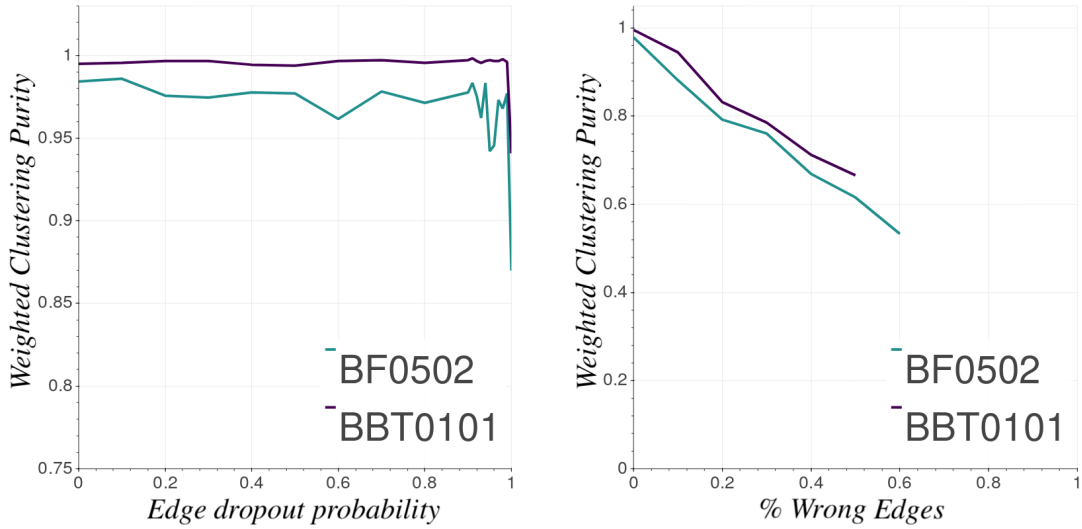


Figure 5.3: Training on BBT-0101 and BF-0502 with increasing edge dropout (left) and added wrong edges (right). Each data point corresponds to one training run for 30 epochs.

achieving 0.52% and 5.55% absolute increase in WCP on BBT-0101 and BF-0502 respectively.

ACCIO. ACCIO provides a more difficult dataset compared to BF and BBT. There are 35 named characters, and many of them never appear on screen at the same time, thus providing no cannot-link edges between these characters. Additionally, the most frequently occurring character is depicted in 30.7% of tracks, whereas the least frequent character in only 0.06% of tracks. The impact of similarity-based edges is compared in Table 5.8. As common in previous works (Sharma et al., 2019a; Zhang et al., 2016b), we evaluate with 36 and 40 clusters for the 35 main character in Table 5.9 and Table 5.10 respectively. Our method significantly outperforms the

Table 5.7: Comparison to state-of-the-art with clustering accuracy (%) at frame-level on both videos: BBT-0101 and BF-0502. Our result is the mean over five runs.

	BBT-0101	BF-0502
ULDML (ICCV '11) (Cinbis et al., 2011)	57.00	41.62
HMRF (CVPR '13) (Wu et al., 2013b)	59.61	50.30
wu2013simultaneous (ICCV '13) (Wu et al., 2013a)	66.77	—
WBSLRR (ECCV '14) (Xiao et al., 2014)	72.00	62.76
VDF (CVPR '17) (Sharma et al., 2017)	89.62	87.46
Imp-Triplet (PacRim '16) (Zhang et al., 2016a)	96.00	–
JFAC (ECCV '16) (Zhang et al., 2016b)	—	92.13
TSiam (FG '19) (Sharma et al., 2019a)	98.58	92.46
SSiam (FG '19) (Sharma et al., 2019a)	99.04	90.87
FINCH (CVPR '19) (Sarfraz et al., 2019)	99.16	92.73
CCL (FG '20) (Sharma et al., 2020)	**99.56**	93.79
FGG	**99.56**	**98.01**

state-of-the-art in unsupervised learning techniques and is comparable to Sarfraz et al. (2019). Note that Sarfraz et al. (2019) is not trainable and He et al. (2018) requires a labeled dataset.

Table 5.8: Influence of similarity-based edges on the ACCIO dataset. *FGG-0* does not add any-similarity based edges. *FGG-100** samples 100% of isolated nodes *before* splitting. This results in OUT_OF_MEMORY, indicated by †. *FGG-100* computes similarity-based edges after splitting, and samples 100% of the remaining isolated nodes. Performance degrades slightly because connecting 3% of the nodes results in relatively more wrong edges compared to BBT and BF.

	#cluster=36		
	P	R	F
FGG-0	0.676	0.755	0.714
FGG-100*	†	†	†
FGG-100	0.654	0.728	0.689

FGG vs. CCL, TSiam and SSiam. In Table 5.11, we report the frame-level clustering performance on all three datasets (as compared to track-level performance in Table 5.2). We observe that FGG outperforms CCL, TSiam, SSiam and also the base features by significant gains.

Table 5.9: Performance comparison on ACCIO with 36 clusters. Score is averaged over 5 runs. We achieve an absolute improvement of 18.5% in B^3 F-Score over the previous state-of-the-art unsupervised learning method.

	#cluster=36		
	P	R	F
IL-HC (AAAI '18) (He et al., 2018)	0.908	0.786	0.843
FINCH (CVPR '19) (Sarfraz et al., 2019)	0.748	0.677	0.711
JFAC (ECCV '16) (Zhang et al., 2016b)	0.690	0.350	0.460
TSiam (FG '19) (Sharma et al., 2019a)	0.749	0.382	0.506
SSiam (FG '19) (Sharma et al., 2019a)	**0.766**	0.386	0.514
CCL (FG '20) (Sharma et al., 2020)	**0.779**	0.402	0.530
FGG	0.654	**0.728**	**0.689**

Computational Complexity. This approach does not scale to arbitrarily large graphs, as the full set of features must be present in memory, and a large adjacency matrix is constructed. We do not optimize our implementation to the highest possible degree, which currently requires the transformation of a dense matrix into a sparse matrix. For this reason, a frame-level evaluation consumes too much memory (see. Table 5.6). However, our sub-track level implementation requires only a reasonable amount of memory even for datasets originating from a feature film such as ACCIO. If required, some more refined implementation tricks could be employed to enable running on larger datasets. See Section 5.4.1 for a discussion on run time. Please note that running on ACCIO takes considerably longer than BF and BBT due to at least 4.5 times more tracks and edges. With the smaller datasets, we can use dense multidimensional tensor multiplications, but due to memory issues we have to use multiple sparse multiplications with ACCIO. This can be resolved once sparse multidimensional tensor multiplications are available in pyTorch.

5.5 Feature Space Projections

We show projections of feature space before the start and after the end of training using t-SNE (van der Maaten and Hinton, 2008) in Figure 5.5 and using Principal Component Analysis (PCA) (Abdi and Williams, 2010) in Figure 5.4. The overlap between clusters as well as the cluster spread is visibly reduced.

Table 5.10: Performance comparison on ACCIO with 40 clusters. Score is averaged over 5 runs. Our method outperforms previous unsupervised methods significantly, with 19% absolute improvement in B^3 F-Score.

	# clusters=40		
	P	R	F
FINCH (CVPR '19) (Sarfraz et al., 2019)	0.733	0.711	0.722
DIFFRAC-DeepID2$^+$ (ICCV '11) (Zhang et al., 2016b)	0.557	0.213	0.301
WBSLRR-DeepID2$^+$ (ECCV '14) (Zhang et al., 2016b)	0.502	0.206	0.292
HMRF-DeepID2$^+$ (CVPR '13) (Zhang et al., 2016b)	0.599	0.23.0	0.332
JFAC (ECCV '16) (Zhang et al., 2016b)	0.711	0.352	0.471
TSiam (FG '19) (Sharma et al., 2019a)	0.763	0.362	0.491
SSiam (FG '19) (Sharma et al., 2019a)	**0.777**	0.371	0.502
CCL (FG '20) (Sharma et al., 2020)	**0.786**	0.392	0.523
FGG	0.663	**0.724**	**0.692**

Table 5.11: Frame-level clustering accuracy (%) of FGG and comparison to CCL (Sharma et al., 2020), TSiam (Sharma et al., 2019a) and SSiam (Sharma et al., 2019a) over all datasets: BBT-0101, BF-0502 and ACCIO.

	#Clusters	Base	TSiam	SSiam	CCL	FGG
BBT-0101	5	94.00	98.58	99.04	**99.56**	**99.56**
BF-0502	6	91.20	92.46	90.87	93.79	**98.01**
ACCIO	36	79.90	81.30	82.00	**83.40**	77.17

5.6 SUMMARY

In this chapter, we have presented a method to obtain improved face feature representations by utilizing the underlying structure of person appearances in video data. A graph is constructed based on this structure, where each node represents a face sub-track and edges indicate either that two tracks belong to the same, or a different person. We exploit the structure as a supervision signal to compute feature representations over each sub-track using graph convolutions, allowing each track's feature to exchange information with its neighbors. The approach creates face-track representations with improved performance for face clustering. We outperform other recent unsupervised approaches on the *Buffy*, *Big Bang Theory* and *ACCIO* datasets significantly. In future work, we would like to interpret the approach as a link prediction problem. Another potential future step is a smarter utilization of negative

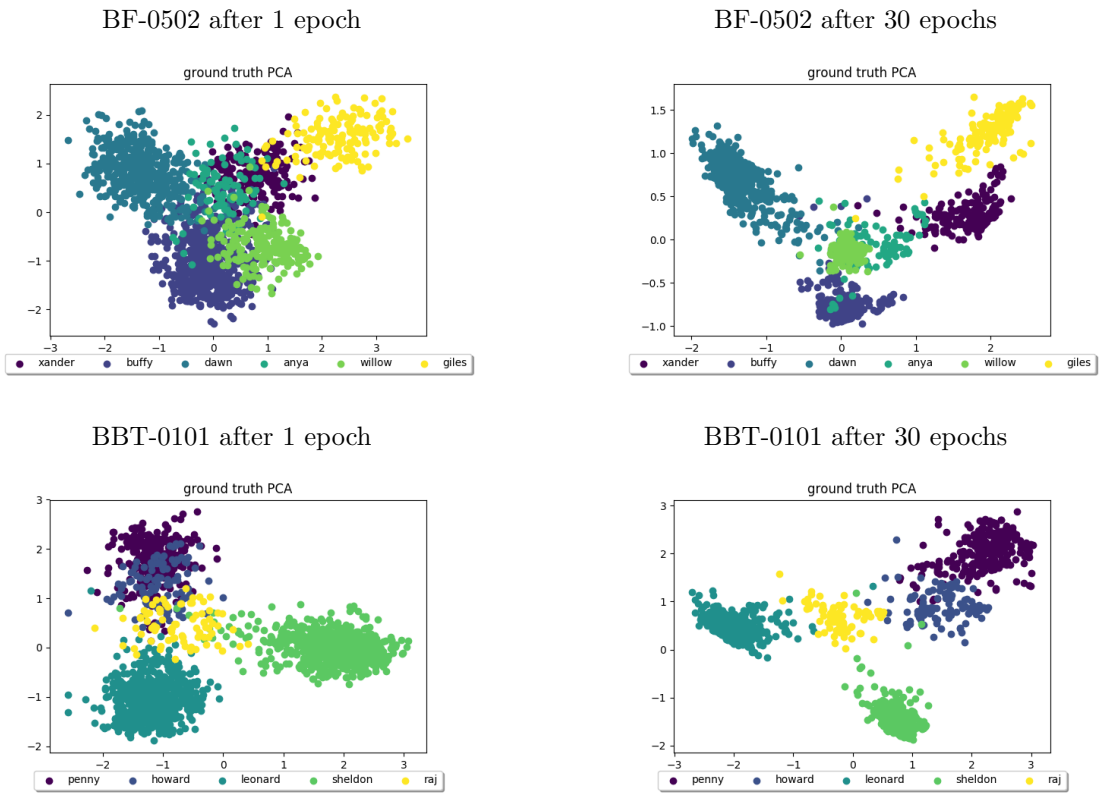
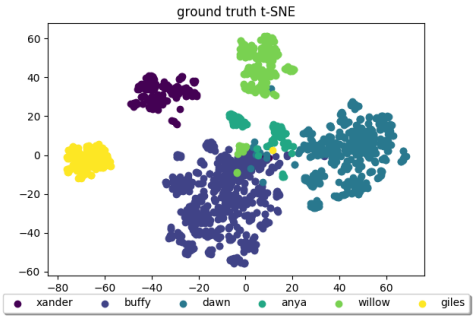


Figure 5.4: Comparison of a PCA projection (Abdi and Williams, 2010) of feature space at the start and end of training on BF-0502 and BBT-0101 using FGG. The ground truth character associations to each data point are color-coded.

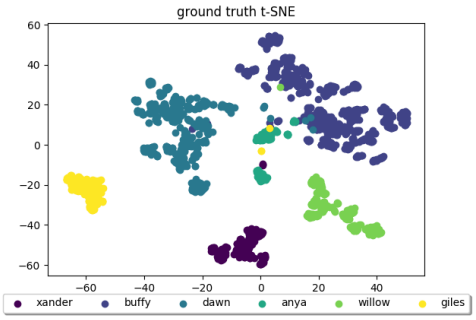
edges (cannot-link), similar to recent work on signed graphs (Derr et al., 2018; Kumar et al., 2016; Yuan et al., 2017).

In the next chapter, we present an alternative strategy to encode the face sub-track in videos into a single descriptor, which we then integrate into our previous method for learning a more powerful representation.

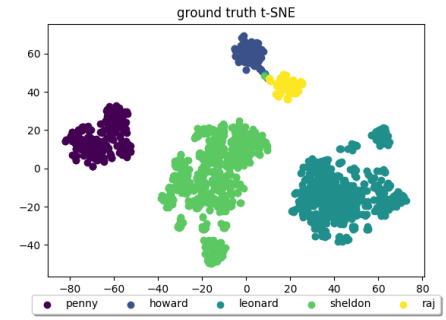
BF-0502 after 1 epoch



BF-0502 after 30 epochs



BBT-0101 after 1 epoch



BBT-0101 after 30 epochs

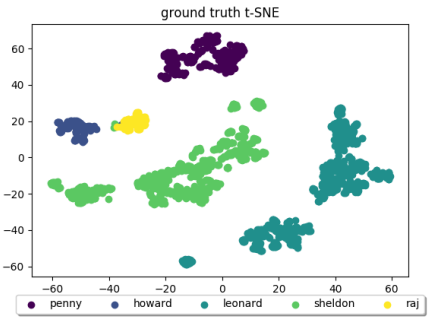


Figure 5.5: Comparison of a t-SNE projections (van der Maaten and Hinton, 2008) of feature space at the start and end of training on BF-0502 and BBT-0101 using FGG. The ground truth character associations to each data point are color-coded.

Chapter 6

Temporal Feature Encoding and Representation Learning

The previous chapters proposed novel methods to generate weak-labels for face representation learning. In this chapter, we study the role of feature encoding, to encode the face sub-track in videos into a single descriptor, which we then use towards learning a more powerful face representation.

Specifically, in this chapter, we first start with presenting a new task of sorting by time a scrambled collection of video clips. Ordering video clips requires us to learn intrinsic characteristics of videos such as how people act, events unfold, and scenes change over time. We propose to learn video ordering using a self-supervised approach and a new feature encoding method that creates a compact feature representation encompassing several modalities including video (images), audio and dialog. We create a new multimodal dataset, namely LSMDC-Ordering for temporal ordering that consists of almost 30K scenes (2-6 clips per scene) from movies that are derived from the Large Scale Movie Description Challenge dataset. The influence of individual and joint modalities are analyzed on two challenging tasks: inferring the temporal order for a set of videos, and multimodal clip retrieval. Our experiments demonstrate that different modalities are indeed complementary and can play an important role in both applications. Following video ordering, as other use cases of feature encoding, we then show that our feature encoding method performs well on action recognition and video face clustering, and achieve gains in performance. The datasets and code are available at *https://github.com/vivoutlaw/tcbp*.

Figure 6.1: Video ordering: Given an unordered collection of video clips, our goal is to infer their correct temporal order by utilizing a compact multimodal feature encoding that exploits high-level semantic concepts such as objects, scenes, dialogues, and sounds in each clip. We visualize five clips (first and last frame) from a scene in our dataset. Can you guess the correct order by considering all modalities?

The content of this chapter is based on the following publication:

- Vivek Sharma, Makarand Tapaswi, and Rainer Stiefelhagen. **"Deep multimodal feature encoding for video ordering"**. In IEEE International Conference on Computer Vision (ICCV): workshop on Large Scale Holistic Video Understanding, 2019. *Oral presentation.*

6.1 Introduction

Temporal sequences are frequently encountered in computer vision problems such as object tracking, video action recognition, video segmentation, video captioning, *etc.*. Compelling advantages of exploiting temporal cues over merely spatial ones have been shown in the recent years (Carreira and Zisserman, 2017; Diba et al., 2017, 2018a, 2019b; Hara et al., 2018). However, such information can only be utilized if the temporal ordering of the data is known.[5]

In this chapter, we consider the task of *ordering video clips* to create a timeline. While ordering a sequence of photos has received much attention (Basha et al., 2012; Dicle et al., 2016; Matzen and Snavely, 2014; Moses et al., 2013; Sadeghi et al., 2015; Schindler et al., 2007), our task is arguably more complex due to motion, variations

[5]The correct order is B - A - D - C - E. Forrest approaches Jenny on the school bus, asks her if he can sit with her, and they introduce each other.

in activities, objects, and context in videos (see Fig. 6.1). To address this problem, we learn a semantic representation for video clips by exploiting high-level concepts from multiple complementary sources – objects, scenes, events, dialog, and activities of interest.

Learning a joint representation for images/videos and text is popular in captioning (Jin et al., 2016; Wu and Han, 2018; Yang et al., 2017b), description (Hori et al., 2017) cross-modal retrieval (Cascante-Bonilla et al., 2019; Guo et al., 2019; He et al., 2019; Peng et al., 2018; Qi et al., 2018; Ren et al., 2019), and summarization (Kim et al., 2014), and is typically achieved using a combination of Convolutional Neural Networks (CNNs) (Carreira and Zisserman, 2017; He et al., 2016) and Recurrent Neural Networks (RNN) (Hochreiter and Schmidhuber, 1997; Kiros et al., 2015). Typical fusion strategies often include concatenation, element-wise product, sum-, average-, or max-pooling, and (self-)attention. Recently, *bilinear pooling* (Fukui et al., 2016; Yu et al., 2017) has gained interest as it efficiently captures pairwise interactions between representations. This is often implemented by using the Tensor Sketch (Gao et al., 2016; Pham and Pagh, 2013) or matrix factorization (Yu et al., 2017) algorithms. Effectively understanding a short video clip (2-5 seconds), involves understanding and capturing interactions between all its modalities - audio, video, and dialog (text). To this end, we propose *Temporal Compact Bilinear Pooling* (TCBP), an extension of the Tensor Sketch algorithm (Pham and Pagh, 2013) that incorporates a temporal dimension. We use TCBP to aggregate features computed by pre-trained multimodal networks over several parts of the video (multiple time-steps) into a compact representation.

Our approach can also be categorized under self-supervised visual representation learning (*e.g.* Fernando et al. (2017); Misra et al. (2016); Sharma et al. (2019a); Wang and Gupta (2015)), that uses video constraints to create millions of positive/negative image pairs (*e.g.* predict the next frame) to learn discriminative image features. Generating such frame-level training data assumes that the video is stationary in a short temporal neighborhood. Complementary to above methods, while ordering clips requires understanding the dynamics of each video, it also affords automatic supervision at the clip-level as a long video (*e.g.* a movie) can be easily segmented into temporally ordered clips.

We highlight our key contributions below:
1) We present temporal ordering of video clips as a holistic video understanding task

that requires learning the dynamics of how events unfold and people act over time. To facilitate this study, we design a new dataset LSMDC-Ordering of almost 30K ordered scenes (each with 2-6 clips) from movies (Sec. 6.4.1).
2) We propose to learn in a self-supervised setting and extend a popular bilinear pooling method to the temporal domain – *Temporal Compact Bilinear Pooling* (TCBP) aggregates multiple temporal segments from a video clip combining audio, video, places, objects, and text (Sec. 6.3).
3) We perform an exhaustive study of the effects of individual and joint modalities on temporal ordering and multimodal clip retrieval. We also demonstrate that TCBP achieves performance gains in action recognition and video face clustering by effectively combining temporal cues.

The remainder of this chapter is structured as follows. Section 6.2 overviews related work. Section 6.3 describes our proposed architecture. Experimental results and their analysis are presented in Sections 6.4. Finally, conclusions are drawn in Section 6.5.

6.2 Related Work

We contrast related topics in self-supervised representation learning, the use of time as supervision, audio-visual learning, and briefly discuss feature encoding methods.

Self-supervised representation learning is increasing in popularity in recent years. This learning paradigm obtains supervision by exploiting the structure within the data, and thus removes the need for an often costly labeling effort. For example, one may learn image representations by exploiting the spatial consistency of images (Doersch et al., 2015), or using a host of spatial/color transforms to train a model that recognizes instances of the transformed images (Dosovitskiy et al., 2014). In a similar vein, face representations are fine-tuned for clustering by creating similar/dissimilar pairs within video sub-structures (Tapaswi et al., 2014b) or a distance matrix (Roethlingshoefer et al., 2019; Sharma et al., 2019a).

Videos are another popular source of learning image representations. Here, tracking an object over several frames can be used to generate similar/dissimilar pairs (Wang and Gupta, 2015). At the frame-level, using temporal consistency within neighboring frames (Misra et al., 2016); or finding an *odd* frame in a triplet of frames (Fernando et al., 2017) are examples of deriving supervision from videos. Perhaps closest to our

work, recently Xu et al. (2019) learn video representations by shuffling a triplet of clips (a similar strategy to (Fernando et al., 2017; Misra et al., 2016)). Their method divides a short clip into three contiguous non-overlapping 16-frame segments, and aims to order these short clips correctly.

Our work differs substantially in scope and technical approach from above approaches. We do not learn image/video representations from scratch, but are interested in learning holistic, multimodal clip representations that can be used to order a variable set of clips (up to 6, but can be extended further) ranging from 2-5 seconds. We use strong pre-trained networks (*e.g.* objects (He et al., 2016), sounds (Aytar et al., 2016)) and combine their information over multiple temporal windows using a new temporal pooling approach.

Time as a supervisory signal. The idea of using temporal continuity or ordering as a signal for supervision is quite popular. For example, Misra *et al.* (Misra et al., 2016) shuffle 3 frames from a video to generate positive/negative sequences and train a model to predict whether the order is correct. However, the use of 3-frame sequences only allows learning image representations. Predicting whether a video flows forwards or backwards is another related task (Pickup et al., 2014; Wei et al., 2018). However, they often require computationally intensive optical flow, and do not encode semantics about objects/scenes. More importantly, only a binary forward/backward ordering is learned, and other shuffled permutations are ignored. The goal of our task is to predict the complete order of *video clips* (up to 6) and not frames, each of 2-5 seconds.

Temporal ordering has also been used for other applications: predicting the future in egocentric videos (Zhou and Berg, 2015), learning the steadiness of visual change in videos (Jayaraman and Grauman, 2016), or to recognize complex, long-term activities (Laxton et al., 2007).

Similar in spirit to our task of ordering of video clips, arranging photos by time has received some attention. In particular, automating the process of creating ordered photo albums from an unordered image collection (Sadeghi et al., 2015); sorting photo collections spanning many years (Matzen and Snavely, 2014; Schindler et al., 2007); or ordering a set of photos taken from uncalibrated cameras (Basha et al., 2012; Dicle et al., 2016; Moses et al., 2013) are all related works. In contrast, we work with an unordered collection of video clips.

Audio-visual learning. Audio-visual speech recognition (Yuhas et al., 1989) is among the earliest examples of using both modalities. More recent works include predicting the sound an object makes upon hitting by observing silent videos (Owens et al., 2016a), or predicting the ambient sound that an image conveys (Owens et al., 2016b). There is also work on learning representations in a joint space. A student-teacher approach is used to transfer the knowledge from visual to audio networks (Aytar et al., 2016); or image captioning is extended to spoken captions (Yusuf Aytar, 2017). Audio-visual representation learning has applications in several tasks such as event classification (Parekh et al., 2018), audio-visual localization (Arandjelović and Zisserman, 2018; Parekh et al., 2018), biometric matching (Nagrani et al., 2018b), sound localization (Arandjelović and Zisserman, 2018; Owens and Efros, 2018), person identification (Azab et al., 2019; Nagrani et al., 2018a; Ngiam et al., 2011; Tapaswi et al., 2012), action recognition (Owens and Efros, 2018), on/off-screen audio separation (Ephrat et al., 2018; Owens and Efros, 2018), and video captioning/description (Hori et al., 2017; Wu and Han, 2018).

Other works in multimodal feature learning transfer knowledge across modalities such as RGB $\rightarrow$ RGB (Hinton et al., 2015), RGB $\rightarrow$ Depth (Gupta et al., 2016), RGB $\rightarrow$ Optical-Flow (Diba et al., 2016; Gupta et al., 2016), RGB $\rightarrow$ Sound (Arandjelovic and Zisserman, 2017), and NIR $\rightarrow$ RGB (Limmer and Lensch, 2016) to learn joint representations in a shared feature space.

In contrast to all these previous works, our goal is to learn a joint video clip representation combining information from audio, images, video, and text sources.

Feature encoding methods have been a mainstay in vision for several decades. Popular approaches before deep learning include bag-of-words (BoW) (Csurka et al., 2004; Sivic and Zisserman, 2003), Fisher vector (FV) encoding (Perronnin et al., 2010) and sparse coding (Yang et al., 2009). In fact, owing to their popularity, FV (Tang et al., 2016), VLAD (Arandjelović et al., 2016; Gong et al., 2014; Miech et al., 2017b), and Bilinear models (Lin et al., 2015; Tenenbaum and Freeman, 2000) with Tensor Sketch (Pham and Pagh, 2013) have also been integrated as specialized layers in neural networks. Such encodings yield promising results on many challenging tasks including action recognition (Diba et al., 2017; Girdhar et al., 2017), and image classification (Chowdhury et al., 2016; Lin et al., 2015). In this work, we adopt

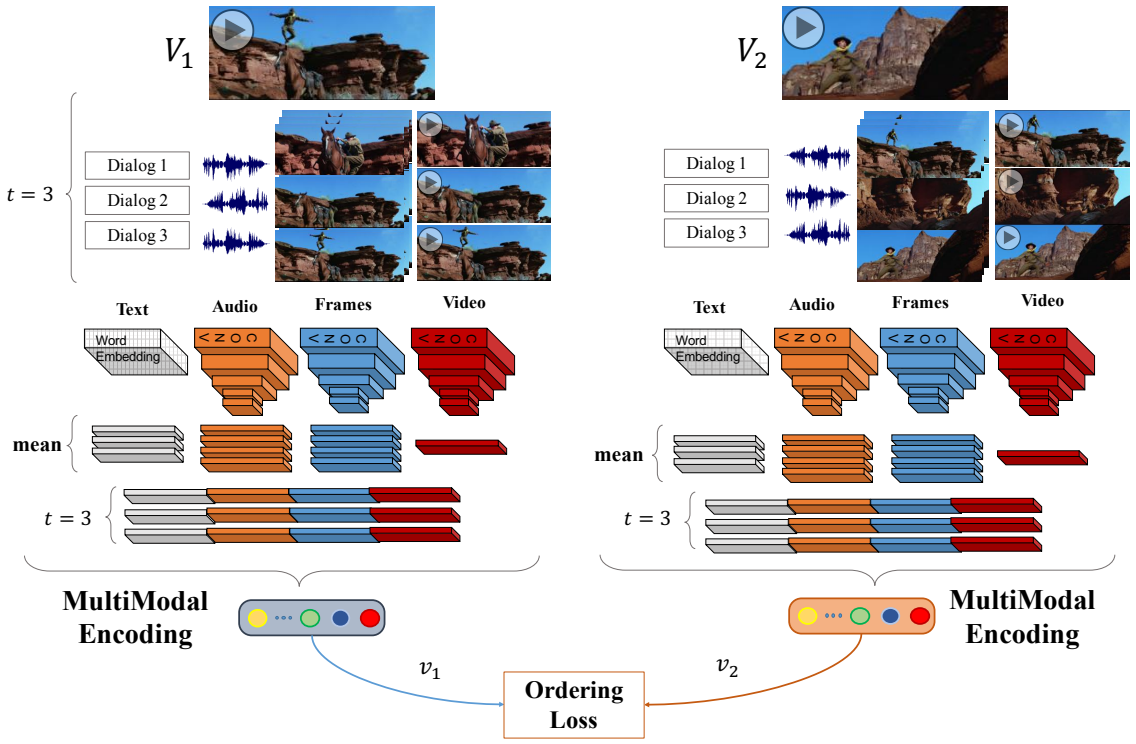


Figure 6.2: Deep Multimodal Feature Encoding. Illustration of the multimodal feature encoding applied for the task of temporal ordering. See Sec. 6.3 for a detailed explanation of the feature learning scheme shown.

bilinear models to encode multimodal information, and extend the Tensor Sketch algorithm to incorporate a temporal dimension.

6.3 Learning to Order Clips

We present our model to obtain multimodal clip representations that can be used to order video clips, as shown in Fig. 6.2. Note that our parameters are learned in a self-supervised approach, as the order of short clips can be inferred automatically by pre-selecting them from a long video.

Deep multimodal video clip representation. For a video clip with N modalities, we compute one output feature map per modality produced by a pre-trained (convolutional) neural network. We denote these feature maps as matrices $\{G_1, G_2, \ldots, G_N\}$, where $G_i \in \mathbb{R}^{c_i \times t}$, c_i denotes the number of channels for the i^{th} modality, and t is

Algorithm 2: Deep Multimodal Feature Encoding Layer

Input: Multimodal Neural Network features for a video clip $\{G_1, G_2, ..., G_N\}$, $G_i \in \mathbb{R}^{c_i \times t}$, where c_i denotes channels of feature maps for each modality, t is the temporal length, and N is the number of modalities.
Output: Encoded feature map $\boldsymbol{v} \in \mathbb{R}^d$ representing the video clip.
Temporal Multimodal Feature Encoding:
1. $X = G_1 +\!\!+ G_2 +\!\!+ \ldots +\!\!+ G_N$, $X \in \mathbb{R}^{c \times t}$, where $c = \sum_{i=1}^{N} c_i$, and $+\!\!+$ is the concatenation operator.
2. $\boldsymbol{v} = EncodingMethod(X)$, $\boldsymbol{v} \in \mathbb{R}^d$, where d denotes the encoded feature dimensions.

the temporal length or number of segments in which the video is divided for feature extraction.

For each clip, we concatenate all feature maps across channels, *i.e.* $X = [G_1, \ldots, G_N], X \in \mathbb{R}^{c \times t}$, where $c = \sum_{i=1}^{N} c_i$. We choose concatenation as it allows us to combine a variable number of modalities, while preserving the complete information obtained from them. The aggregated feature X is encoded using some method $E : X \rightarrow \boldsymbol{v}$, to compute the multimodal representation $\boldsymbol{v} \in \mathbb{R}^d$, where d is the dimensionality of the final embedding space. Algorithm 2 sketches the steps of the proposed multimodal feature encoding.

6.3.1 Temporal Compact Bilinear Pooling

We adopt bilinear pooling to learn multimodal representations as it possesses the ability to capture interactions between each element of the feature representation with one-another (much like a polynomial kernel in SVMs (Burges et al., 1998)). Additionally, bilinear pooling methods have been shown to work well for related tasks of image classification (Chowdhury et al., 2016; Gao et al., 2016; Lin et al., 2015), video classification (Diba et al., 2017) and visual-question answering (Fukui et al., 2016). We briefly discuss bilinear models, Tensor Sketch projection and Compact Bilinear Pooling (CBP), and then propose our extension for handling temporal data.

Preliminaries. A bilinear pooling function is an operator on the outer product of a vector $\boldsymbol{x} \in \mathbb{R}^c$ to obtain $\boldsymbol{v} \in \mathbb{R}^d$ such that:

$$\boldsymbol{v} = W[\boldsymbol{x} \otimes \boldsymbol{x}^T], \tag{6.1}$$

where $\otimes$ denotes the outer product, $[\cdot]$ turns the matrix into a vector by concatenating the columns, and $W \in \mathbb{R}^{d \times c^2}$ are model parameters. However, as feature dimensions are often big (*e.g.* $c = 1024$), the number of parameters W is in the billions (assuming $d = 1024$ as well), making learning challenging.

To alleviate this problem, Gao et al. (2016) show how the high-dimensional second-order function can be approximated by a low dimensional projection function using the Tensor Sketch algorithm (Pham and Pagh, 2013). This removes the need for computation of the expensive outer product and obtains the final vector directly.

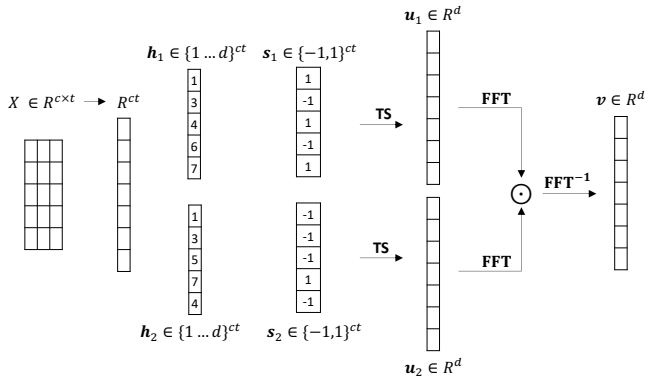
Tensor Sketch (TS) (Pham and Pagh, 2013) also called the Count Sketch, projects a vector $\boldsymbol{x} \in \mathbb{R}^c$ to $\boldsymbol{v} \in \mathbb{R}^d$. Two sets of auxiliary vectors $\boldsymbol{h}$ and $\boldsymbol{s}$ are initialized randomly from a uniform distribution (and held fixed thereafter) to perform this projection $\boldsymbol{u}_1 = \psi(\boldsymbol{x}, \boldsymbol{h}_1, \boldsymbol{s}_1)$ and $\boldsymbol{u}_2 = \psi(\boldsymbol{x}, \boldsymbol{h}_2, \boldsymbol{s}_2)$. The sign vector $\boldsymbol{s} \in \{-1, 1\}^c$ indicates whether the element in $\boldsymbol{x}$ will be added or subtracted from the final value at a location determined by $\boldsymbol{h} \in \{1, \ldots, d\}^c$. In particular, for every element $\boldsymbol{x}[i]$, its destination in $\boldsymbol{u}$ is given by $j = \boldsymbol{h}[i]$, and the value is accumulated as $\boldsymbol{u}[j] = \boldsymbol{u}[j] + \boldsymbol{s}[i] \cdot \boldsymbol{x}[i]$. Finally, $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$ (corresponding to $\boldsymbol{h}_1, \boldsymbol{s}_1$ and $\boldsymbol{h}_2, \boldsymbol{s}_2$) are convolved with each other to compute $\boldsymbol{v}$ by performing element-wise multiplication in the Fourier domain.

Compact Bilinear Pooling (CBP) (Gao et al., 2016). When working with images, a CNN feature map X often preserves the encoding of spatial locations to yield $X \in \mathbb{R}^{c \times s}$, where c is number of channels (as in the above discussion) and s refers to spatial locations (typically, $h \times w$). To work with such features, (Gao et al., 2016) first performs TS projection on the vector at each spatial element s, followed by sum pooling.

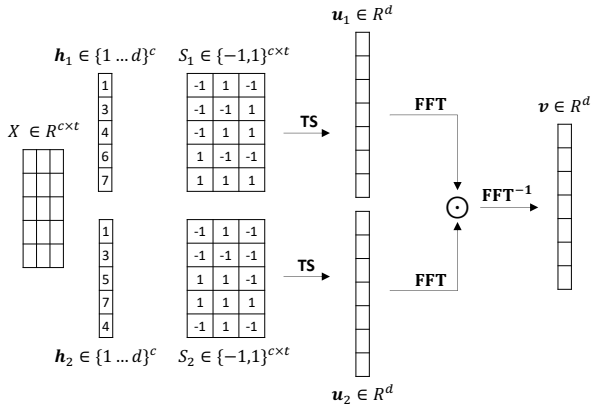
$$X \in \mathbb{R}^{c \times s} \xrightarrow{\text{TS projection}} \mathbb{R}^{d \times s} \xrightarrow{\text{sum pool over } s} \boldsymbol{v} \in \mathbb{R}^d \,. \tag{6.2}$$

For more details please refer to (Gao et al., 2016).

Temporal Compact Bilinear Pooling (TCBP). We now present our extension of the Tensor Sketch projection algorithm to incorporate a temporal dimension. Recall, our feature $X \in \mathbb{R}^{c \times t}$ is the result of stacking features from all modalities. The key difference between CBP and TCBP is the process in which the TS projection vector $\boldsymbol{h}$ and matrix S are created (see Fig. 6.3). In particular, we initialize $\boldsymbol{h} \in \{1, \ldots, d\}^c$ as a c dimensional vector, while $S \in \{-1, 1\}^{c \times t}$ as a matrix. Choosing $\boldsymbol{h}$ independent



(a) Compact Bilinear Pooling (CBP)-Flat



(b) Temporal Compact Bilinear Pooling (TCBP)

Figure 6.3: CBP-Flat vs. TCBP. In this toy setup, we initialize random vectors $\boldsymbol{h}$, $\boldsymbol{s}$ and matrix S such that $c = 5$, $d = 7$, and $t = 3$. In CBP-Flat, $\boldsymbol{h}$ has different values across the temporal dimension, while in TCBP, $\boldsymbol{h}$ does not depend on t. This ensures that Tensor Sketch projects features across time to the same output index. See Sec. 6.3.1 for a detailed explanation.

of time, ensures that a feature index i in X ($X[i,t]$) is always encoded to the same destination $j = \boldsymbol{h}[i]$, albeit with a different sign. Similar to TS, we can write $j = \boldsymbol{h}[i]$ (independent of t), while, $\boldsymbol{u}[j] = \boldsymbol{u}[j] + \sum_t S[i,t] \cdot X[i,t]$. The TS projection is followed by the convolution in Fourier domain. Algorithm 3 presents a summary of the above approach.

$$X \in \mathbb{R}^{c \times t} \xrightarrow{\text{TCBP}} \boldsymbol{v} \in \mathbb{R}^d . \tag{6.3}$$

We compare CBP and TCBP with regards to number of parameters and computational efficiency in Table 6.1. We wish to assert that the use of both CBP as well as TCBP for multimodal feature encoding is novel. We learn a projection function to model multimodal data that includes text, images, audio and video.

Algorithm 3: Temporal Compact Bilinear Pooling

Input: Multimodal feature map $X \in \mathbb{R}^{c \times t}$, where c and t are the number of channels and temporal segments.
Output: Multimodal encoded feature map $\boldsymbol{v} \in \mathbb{R}^{d}$, where d is the encoded feature dimension.
Procedure:
1. Initialize random vector $\boldsymbol{h}_k \in \{1, \ldots, d\}^c$, and $S_k \in \{-1, 1\}^{c \times t}$ from a uniform distribution, $k = 1, 2$.
2. Compute the count sketch projection function $\boldsymbol{u} = \Psi(X, \boldsymbol{h}, S) = \{\boldsymbol{u}_1, \ldots \boldsymbol{u}_d\}$, where $\boldsymbol{u}_j = \sum_{i:\boldsymbol{h}[i]=j} \sum_t S[i,t] \cdot X[i,t]$.
3. Finally, compute the output encoded vector as $\boldsymbol{v} = FFT^{-1}(FFT(\Psi(X, \boldsymbol{h}_1, S_1)) \circ FFT(\Psi(X, \boldsymbol{h}_2, S_2)))$, where $\circ$ denotes element-wise multiplication operator.

Table 6.1: TCBP encodes several temporal segments without much additional overhead. Parameters c, t, d denote the number of channels, temporal segments, and the projected dimension.

	CBP	TCBP
Input dimensions	$\mathbb{R}^c$	$\mathbb{R}^{c \times t}$
Output dimensions	d	d
Parameters (h, s)	$2 \cdot 2c$	$2 \cdot (c + ct)$
Computation	$O(c + d \log d)$	$O(ct + d \log d)$

CBP-Flat. Note that TCBP is not the same as applying CBP directly on a flattened vector $X \in \mathbb{R}^{c \times t} \rightarrow \mathbb{R}^{ct}$, as this would require creating $\boldsymbol{h} \in \{1, \ldots, d\}^{ct}$ and $\boldsymbol{s} \in \{-1, 1\}^{ct}$ (see Fig. 6.3). We empirically show that TCBP also outperforms CBP-Flat.

6.3.2 Learning via Temporal Ordering

We learn a multimodal representation for video clips using temporal ordering. For example, consider a movie scene that consists of M video clips, $(V_1, \ldots, V_M)$. These clips are ordered by the content creators to tell a story in the most natural way, and often encode principles such as causality (*e.g.* pushing a car gets it rolling) or temporal progression (*e.g.* enter the house before removing coat).

Training. We randomly sample an ordered consecutive pair of clips (V_i, V_j) from the scene and train our model to learn that V_i appears *before* V_j, denoted as $V_i \succ V_j$. We use the order violation error (Vendrov et al., 2016) between a pair of ordered clips:

$$E(V_i, V_j) = \| \max(0, \phi(V_i) - \phi(V_j)) \|^2 , \tag{6.4}$$

where $\phi(V) \in \mathbb{R}^D_+$ is the clip representation used for temporal ordering, and the loss encodes feature $\phi(V_i)$ to appear "before" $\phi(V_j)$ (*e.g.* below and to the left of in the 2D positive subspace). As the error function has a trivial solution (learning all representations as 0 vectors), we also include unordered (negative) pairs in our formulation. In particular we learn the parameters of $\phi(\cdot)$ by minimizing

$$\mathcal{L} = \sum_{(V_i,V_j)\in\mathcal{O}} E(V_i, V_j) + \sum_{(V_i,V'_j)\in\mathcal{U}} \max(0, \alpha - E(V_i, V'_j)) , \tag{6.5}$$

where $\mathcal{O}$ is the set of all ordered pairs, $\mathcal{U}$ contains unordered pairs, and α represents an expected margin of separation between representations of unordered video clips. We implement unordered pairs by selecting a different clip V'_j from the same batch as training ordered pairs.

Inference. To order the clips during inference, we first encode them as $\phi(V)$ and compute ordering error $E(V_i, V_j)$ between all pairs of clips in the scene. Note that, there are $M!$ possible ways to sort M unordered clips. We use a simple brute-force approach and pick the sequence that results in the smallest overall pairwise ordering error. While this approach does not scale well, it is an acceptable solution when the maximum number of clips in a scene is limited, $M_{\max} = 6$. We leave exploration of other sorting/ranking techniques (*e.g.* Doughty et al. (2019) regresses one value) for the future. We treat a collection of clips as correctly sorted only when the predicted order fully matches the ground-truth order.

6.3.3 Implementation details

We present clip representation and training details below.

Sampling temporal segments from a clip. We treat 16 contiguous frames of a video clip as a temporal segment (commonly used in spatio-temporal networks). While most clips from our dataset have 4 to 6 segments, we use 3 segments for

most experiments to incorporate some variation in selected segments during training. As TCBP depends on the number of temporal segments (recall $S \in \{-1, 1\}^{c \times t}$), we choose to use $t = 3$ segments to represent all clips. We consider three segment sampling strategies: S-Random chooses three contiguous segments randomly; S-First picks the first three segments; and S-Last picks the last three segments.

Multimodal representations from pre-trained models. We extract a variety of features for each temporal segment of a video. (1) Object (I) features are extracted using a ResNet50 pre-trained on the ImageNet dataset (Russakovsky et al., 2015); (2) Place/Scene (P) features are obtained using a ResNet50 pre-trained on the Places365 dataset (Zhou et al., 2017); (3) Video/Activity (R) features are obtained using a 3D ResNet50 (Hara et al., 2018) pre-trained on the Kinetics-400 dataset (Carreira and Zisserman, 2017); (4) Dialog/Subtitle (S) features are computed using Glove6B (Pennington et al., 2014) pre-trained word vectors; and (5) Audio (A) features are extracted using the SoundNet (Aytar et al., 2016) model pre-trained on Flickr videos.

Object (I) and place/scene (P) frame-level representations are obtained from the `avgpool` layer of ResNet50, and are in $\mathbb{R}^{2048}$. We additionally mean pool across 16 frames of the temporal segment. To compute a video (R) representation, we reshape the temporal segment to $112 \times 112 \times 16$, and extract `avgpool` features from our 3D ResNet50 resulting in $\mathbb{R}^{2048}$. We encode the audio signal (A) corresponding to the temporal segment and compute `pool5` features, from the SoundNet model. Averaging across channels results in a feature in $\mathbb{R}^{256}$. Our text modality (S) consists of transcribed dialog, closed captions, or subtitles, and *not* manually curated video captions. We use word embeddings from Glove6B (Pennington et al., 2014), and mean pool all words in a clip to obtain the feature map in $\mathbb{R}^{300}$. As splitting words across temporal segments is not trivial, we use the same feature for all temporal segments.

Clip representations. We use the same network architecture for both CBP and TCBP. The input $X \in \mathbb{R}^{c \times t}$ is a concatenation of modalities discussed above. We first apply a linear layer W_1 to obtain $\bar{X} \in \mathbb{R}^{2048 \times t}$. Encoding this with CBP/TCBP[6]

[6] In practice, with several design choices we found that $d = 8,192$ was sufficient for reaching the best performance, and that increasing the Tensor Sketch (TS) projection dimension d further more had a very small boost in performance. The projection parameter for the TS was optimized efficiently for our task.

results in $\boldsymbol{v}_{\text{enc}} \in \mathbb{R}^{8192}$. We perform feature normalization via a signed square-root operation followed by ℓ_2 normalization. We add a second linear layer W_2 to obtain a generic clip representation that may be used for any application, $\boldsymbol{v}_{\text{clip}} \in \mathbb{R}^{4096}$. Finally, we compute the temporal ordering feature with a third linear layer as $\phi(V) = |W_3(\text{ReLU}(v_{\text{clip}}))|$. The absolute value $\phi(V) \in \mathbb{R}_+^{2048}$, is used to compute the ordering error. All linear layers have biases, but are omitted for brevity.

Training. Our model is implemented in PyTorch v0.4. We use a batch of 32 samples and update parameters with SGD: 10^{-3} learning rate, 0.9 momentum, and weight decay 5×10^{-4} for 5K-8K iterations. We train the model with S-Random sampling strategy as a form of data augmentation.

6.4 Evaluation

In this section, we first introduce our dataset for video clip ordering. Then, we present experiments on inferring the temporal order for a set of scrambled clips, and a multimodal clip retrieval task. Finally, we also show how TCBP can be applied to action recognition and video face clustering.

6.4.1 Dataset

We create a new dataset LSMDC-Ordering to train and evaluate our task of temporal ordering videos. We choose to use clips from the *Large Scale Movie Description Challenge* (LSMDC) (Maharaj et al., 2017; Rohrbach et al., 2017b; Torabi et al., 2016) dataset as they are publicly available, contain several modalities, and are sourced from movies (stories) that usually care about temporal order. The LSMDC dataset consists of 202 movies and 118,081 video clips (2-5 seconds). Each clip is associated with an Audio Description (AD) that narrates the visual content. However, as we are interested in applying our model to any video, we use subtitles (spoken dialog), and AD are not part of our text modality. Unfortunately, having clips with AD implies that they contain little to no dialog.

Scene boundary detection. We group temporally contiguous clips into a *scene* – a set of clips that occur in one place, or contain a group of related events. The clips often have small (few seconds) to large (few minutes) gaps between them. Clips

Table 6.2: Number of scenes in our dataset with 2-6 clips.

Scene size	2	3	4	5	6	2-6
Training	13455	6711	3097	1382	624	**25269**
Validation	958	472	203	100	51	**1784**
Test	1333	588	325	135	62	**2443**

within a scene are used for the ordering task. Detecting scenes is often preceded by detecting shots. However, as the LSMDC dataset contains pre-segmented clips, we redefine the problem as grouping clips into unique scenes. We first use a dynamic programming algorithm (Tapaswi et al., 2014a) to assign clips to scenes. However, scenes created using this method are often quite large and contain between 15-30 clips. We further segment the scenes into shorter sequences (2-6 clips) using the FINCH clustering algorithm (Sarfraz et al., 2019), and manually verify the data for consistency.

We use the training, validation, and test splits, as used in the LSMDC captioning task. In total, there are almost 30K scenes over all splits. In Table 6.2, we present the number of scenes that have 2 to 6 clips in each split. See Fig. 6.4 for some example scenes of variable length from our dataset. Approximately 65% of clips have 4 temporal segments (of 16 frames at 24fps), and there are no clips with less than 48 frames (*i.e.* with $t = 3$).

6.4.2 Temporal Ordering

We present an ablation study evaluating various design choices, followed by a comparison against baselines. We use *ordering accuracy*, a strict metric that requires all clips to be sorted in the correct sequence. When not specified otherwise, the encoding is CBP with $t = 3$ and S-Last sampling. Methods are trained with the ordered component of the loss, $(V_i, V_j) \in \mathcal{O}$ (first term in Eq. 6.5).

Study of modalities and sampling strategy. Table 6.3 reports performance of using individual and joint modalities over three different clip sampling options. For the individual modalities, we do not perform channel reduction, however do include the W_2 and W_3 linear layers (see impl. details). Audio (A), Places (P), and ImageNet (I) modalities seem important for this task, while the joint PI and

Table 6.3: Temporal ordering performance with different $t = 3$ segment sampling strategies and various modality combinations: A: Audio, P: Places, I: Objects, R: Video, and S: Subtitles/Text.

	Modality	S-First	S-Last	S-Random	Chance
Validation	A	22.90	57.97	57.21	31.78
	P	26.40	76.35	73.93	31.78
	I	22.42	77.32	74.57	31.78
	R	39.13	28.08	26.90	31.78
	S	23.79	22.74	21.70	31.78
	PI	30.16	**79.65**	71.86	31.78
	API	24.56	**78.70**	73.15	31.78
	APIS	21.47	77.39	71.33	31.78
	APIR	17.04	77.24	46.92	31.78
Test	PI	34.10	81.13	79.39	31.89
	API	39.13	**85.06**	**84.57**	31.89
	APIS	38.41	83.24	82.11	31.89
	APIR	25.01	82.48	79.38	31.89

API feature representations perform best. It is interesting to see that the video features (APIR) and text embedding (APIS) reduce performance as compared to API. This may be as motion features encoded in R are not strong enough to predict what happens over larger temporal ranges, while the absence of subtitles hurts the text modality. Among the sampling strategies, we see that S-Last performs best, possibly since it contains information closest to the next clip.

Impact of number of clips in a scene. Recall, our dataset consists of scenes with 2-6 clips. In Table 6.4, we present ordering accuracy on subsets of scenes that contain an equal number of clips. While PI and API are similar for scenes with 2-4 clips, API works much better for scenes with more clips (5 or 6). Analyzing failures, we see that in stationary scenes, PI fails to accurately predict the future, while audio may provide some help. We choose API as the default modality for all further experiments.

Impact of number of temporal segments. Recall each segment t represents a chunk of 16 frames, and clips in our dataset range from 4 to 11 segments. Table 6.5 confirms our hypothesis that using more segments leads to better clip representations improving performance for both modalities.

Figure 6.4: Examples scenes from the dataset with 2-4 clips.

CBP vs. TCBP. Table 6.6 compares the impact of our encoding approach with increasing number of temporal segments. In particular, CBP and TCBP are identical when $t = 1$, while TCBP shows improvements over CBP for $t = 2$ and 3 as it is better able to encode longer clips, while CBP uses sum pooling. We also compare TCBP against CBP-Flat that uses flattened ct-dimension. At $t = 3$, the ordering accuracy on the test set with CBP-Flat is 84.37%, while TCBP achieves 86.23%.

Comparing TCBP against other encoding baselines. We compare TCBP with several popular pooling methods. As before, we start with an input feature map $X \in \mathbb{R}^{c \times t}$. Specifically, in Table 6.7, we compare TCBP against variants of method to exploit the temporal dimension effectively, they are ConcatT-MLP, Mean-Pool and NetVLAD. Now, we briefly explain the architecture for each pooling method. **ConcatT-MLP** first applies the linear layer to obtain $\bar{X} \in \mathbb{R}^{2048 \times t}$. We then concatenate the temporal segments to form a vector in $\mathbb{R}^{2048 \cdot t}$. **Mean-Pool** averages the feature across temporal segments t, resulting in a vector in $\mathbb{R}^{c}$. **NetVLAD** first

Table 6.4: Ordering accuracy for varying number of clips per scene in the validation split.

# Clips in scene	2	3	4	5	6	2-6
# Samples	958	472	203	100	51	1784
Random	50.00	16.67	4.17	0.83	0.14	31.78
PI	96.86	60.38	91.13	22.00	1.96	**79.65**
API	95.51	56.14	87.19	31.00	31.37	**78.70**
APIS	98.28	54.01	76.34	16.45	25.13	77.39
APIR	100	42.58	88.18	40.00	0.0	77.24

Table 6.5: Validation split ordering accuracy by changing the number of temporal segments used to represent video clips.

PI			API		
$t=1$	$t=2$	$t=3$	$t=1$	$t=2$	$t=3$
71.14	77.30	**79.65**	54.88	**77.92**	**78.70**

applies a linear layer with 512 output dim, resulting in a feature map in $\mathbb{R}^{512\times t}$. This is fed to NetVLAD (Arandjelović et al., 2016; Girdhar et al., 2017; Miech et al., 2017b) with 32 clusters[7] resulting in a feature vector of $\mathbb{R}^{16,384}$. All above methods are followed by two fully-connected layers resulting in a vector in $\mathbb{R}^{1024}$ that is used for the ordering loss.

Table 6.7 shows that TCBP outperforms all methods, including strong baselines of CBP and NetVLAD. In fact, ConcatT-MLP performs very poorly, while simple Mean-Pool shows limited performance. We expect small improvements for TCBP over CBP as the main difference lies in the projection of temporal segments directly instead of pooling over time after projection. However, we believe that TCBP is a principled way to handle multiple temporal segments within the bilinear pooling framework.

Role of negative mining. We now analyze the impact of also including unordered examples during training. We observe that including unordered/negative pairs consistently improves the performance for all methods (Table 6.8). However, contrary to our initial expectation, it seems that we do not necessarily need negative examples.

[7]We optimized NetVLAD hyperparameters by evaluating #clusters={16,32,64} × feat-dim={256,512,1024}, and found #clusters=32 and feat-dim=512 to provide the best performance.

Table 6.6: Ordering performance with CBP and TCBP. More temporal segments increases the gap between the two methods.

	Validation			Test		
Method	$t=1$	$t=2$	$t=3$	$t=1$	$t=2$	$t=3$
CBP	54.87	77.92	78.70	68.65	84.69	85.06
TCBP	54.87	78.13	**79.43**	68.65	85.26	**86.23**

Table 6.7: Comparing TCBP with other popular encoding strategies.

Method	Validation	Test
Chance	31.78	31.89
ConcatT-MLP	19.78	37.17
Mean-Pool	66.59	69.58
NetVLAD	68.11	83.79
CBP	78.70	85.06
TCBP	**79.43**	**86.23**

6.4.3 Multimodal Retrieval

We present a second task that showcases the effectiveness of our video representation. Given a query clip, our goal is to find other clips that are from the same scene.

Dataset. We only consider the test set that has 2,443 scenes. We use at least one clip (up to 25%) from each scene as a query, and the rest are treated as gallery for retrieval. In total, we obtain 2,770 clips as queries and 4,466 as gallery.

Metrics. Retrieval performance is measured using mean Average Precision (mAP) at various ranks.

Method. We do not train new models for this task, but analyze the quality of clip representations ($\boldsymbol{v}_{\text{clip}}$ in Sec. 6.3.3) learned from the temporal ordering task. In particular, we represent a video clip by taking 3 segments at a time and encode them with TCBP, followed by W_2. For clips longer than 3 segments, we average pool multiple non-overlapping 3 segment chunks, to obtain a final 4096-d representation.

As baselines, we compare against individual modalities by average pooling base features obtained from our pre-trained models. We also compare against a late score fusion of modalities PI (LF-PI) and API (LF-API).

Table 6.8: Role of negative mining for temporal ordering.

Negatives?	NetVLAD		CBP		TCBP	
	Val	Test	Val	Test	Val	Test
No Neg.	68.11	83.79	78.70	85.06	79.43	**86.23**
With Neg.	71.21	84.99	84.59	88.95	83.18	**89.19**

Table 6.9: Multimodal retrieval results. The query set consists of 2770 clips, and the test consists of 4466 clips, from 2443 scenes. TCBP with negatives.

Methods	mAP	P1	P5	P10	P50
A	1.65	1.73	3.17	4.22	9.27
P	68.49	73.93	88.66	91.80	95.81
I	72.30	81.22	91.40	93.79	96.96
LF-PI	73.53	80.54	92.22	94.25	97.50
LF-API	72.39	79.31	91.73	93.68	97.15
TCBP-API (Ours)	**74.61**	**82.52**	**94.35**	**95.75**	**98.43**

Results. Table 6.9 shows that our multimodal feature representation using TCBP outperforms late fusion models as well as individual modalities. Interestingly, we see that LF-API performs worse than LF-PI, indicating that fusion of multiple modalities is not trivial. We argue that a better encoding method like TCBP helps obtain an effective multimodal representation. Fig. 6.10 shows qualitative retrieval results for LF-API and TCBP-API. In particular, we highlight examples where TCBP-API seems to be more effective.

6.4.4 Action Recognition

In our final experiment, we evaluate the effectiveness of TCBP for video action recognition.

Dataset. We evaluate TCBP on two action and activities datasets: HMDB51 (Kuehne et al., 2013) (51 actions, 6,766 clips) and UCF101 (Soomro et al., 2012) (101 actions, 13,320 clips). We report average accuracy over the pre-defined splits for both datasets.

Method. We use TLE (Diba et al., 2017) as the base architecture, and employ TCBP as the encoding method. In particular, we use the C3D ConvNet (Tran et al., 2015) pre-trained on the Sport-1M dataset (Karpathy et al., 2014). The network

Table 6.10: Qualitative top-3 retrieval results for a given query.

Table 6.11: C3D ConvNets. Comparison of accuracy (%) of TCBP with C3D ConvNet against state-of-the-art methods over all three splits of UCF101 and HMDB51.

Method	UCF101	HMDB51
SpatioTemporal ConvNet (CVPR '14) (Karpathy et al., 2014)	65.4	–
LRCN (CVPR '15) (Donahue et al., 2015)	82.9	–
Composite LSTM Model (ICML '15) (Srivastava et al., 2015)	84.3	44.0
iDT+FV (ICCV '13) (Wang and Schmid, 2013)	85.9	57.2
Two Stream (NIPS '14) (Simonyan and Zisserman, 2014a)	88.0	59.4
C3D (ICCV '15) (Tran et al., 2015)	82.3	56.8
TLE: Bilinear+TS (CBP) (CVPR '17) (Diba et al., 2017)	85.6	59.7
TLE: Bilinear (CVPR '17) (Diba et al., 2017)	86.3	60.3
TLE: TCBP (ours)	**88.03**	**61.58**

operates on a stack of 16 RGB frames of size 112×112. Following the setup of Diba et al. (2017), we employ TSN (Wang et al., 2016) architecture with 3 segments pushed through the C3D ConvNet. Same as (Diba et al., 2017), we extract the convolutional feature maps from the last convolutional layers and feed them as input to TCBP. This is followed by a classification layer with a K-way softmax, where K is the number of action categories. Note that convolutional feature maps from each segment of the last layer of C3D produce an output of size $\mathbb{R}^{512\times2\times7\times7}$. Three such segments are first aggregated via element-wise multiplication (similar to TLE), and followed by TCBP projection with $t = 2$ resulting in a vector of $\mathbb{R}^{8192}$ dimensions. We use the same training and evaluation setup as (Diba et al., 2017) for a fair comparison.

Results. Table 6.11 shows the performance of TCBP with C3D ConvNets compared against several other methods. The goal of this experiment is to show that TCBP can improve the performance of the base C3D (Tran et al., 2015). It is also interesting to see that, TCBP improves the C3D ConvNet performance over the two-stream ConvNets (Simonyan and Zisserman, 2014a) on both datasets. Further, TCBP outperforms Bilinear pooling, CBP (Diba et al., 2017), and iDT+FV (Wang and Schmid, 2013) by a significant margin. We believe that TCBP outperforms other methods due to its ability to effectively encode temporal cues in the video, something that other methods (Diba et al., 2017; Simonyan and Zisserman, 2014a; Tran et al., 2015; Wang and Schmid, 2013) do not. Note that, one would expect to obtain better performance on any task when using features from a newer model (*e.g.* Hara et al. (2018)). For a fair comparison with (Diba et al., 2017; Tran et al., 2015) we resort

to C3D features. It is good to see that TCBP can also improve performance when using C3D. In practice, NetVLAD has been shown to perform worse than Bilinear pooling (*e.g.* CBP) in (Diba et al., 2017; Girdhar et al., 2017). This is most likely due to CBP capturing interactions between features at each channel, thus leading to a strong representation.

6.4.5 Video Face Clustering

In our last experiment, we evaluate the effectiveness of TCBP for video face clustering task.

Datasets and metric. We present our evaluation on three challenging datasets: BBT, BF and ACCIO[8], discussed in Section 2.3. We summarize the statistics of datasets in Table 2.1. In terms of evaluation metric following the previous works, for BBT and BF we report clustering accuracy (ACC or WCP), and for ACCIO in addition to ACC, we report BCubed Precision (P), Recall (R) and F-measure (F), as discussed in Section 2.4.

Method. We adopt the VGG-2 face CNN (Cao et al., 2018), a ResNet50 model, pre-trained on MS-Celeb-1M (Guo et al., 2016) and fine-tuned on 3.31 million face images of 9,131 subjects (VGG2 data). Input RGB face images are resized to 224×224, and pushed through the CNN. We extract `pool5_7x7_s1` features, resulting in $\mathbb{R}^{2048}$.

In particular, we employ TCBP as the encoding method to train our Siamese network with track-level supervision (TSiam). Note that we stack feature maps for 5 consecutive frames $\mathbb{R}^{2048\times5}$, and then encode them via TCBP projection with $t = 5$ resulting in a vector of $\mathbb{R}^{8192}$ dimensions, followed by fully-connected layers $\mathbb{R}^{512}$ and $\mathbb{R}^{2}$. The full network comprises as ($\mathbb{R}^{2048\times5} \rightarrow \mathbb{R}^{8192} \rightarrow \mathbb{R}^{512} \rightarrow \mathbb{R}^{2}$). Note that the last linear layer is part of the contrastive loss, and we use the feature representations at $\mathbb{R}^{512}$ for clustering. For a fair comparison, we use the same network architecture with TCBP for all methods. The network is trained using the contrastive loss, and parameters are updated using Stochastic Gradient Descent (SGD) with a fixed learning rate of 10^{-3}. Since the labels are obtained automatically for each video, overfitting is not a concern. For model training, we decompose each tracklet

[8]Note that, in accordance with previous literature (Zhang et al., 2016b), we use 36 and 40 clusters for the 35 main characters.

Table 6.12: Clustering accuracy computed at track-level on the training episodes, with a comparison to all evaluated models. † indicates OUT_OF_MEMORY.

	BBT-0101	BF-0502
Base	0.932	0.836
TSiam (FG '19) (Sharma et al., 2019a)	0.964	0.893
SSiam (FG '19) (Sharma et al., 2019a)	0.962	0.909
CCL (FG '20) (Sharma et al., 2020)	0.982	0.921
FGG (ACMMM '19) (Roethlingshoefer et al., 2019)	**0.996**	**0.980**
with TCBP		
TSiam	0.972	0.912
SSiam	0.976	0.919
CCL	**0.991**	**0.935**
FGG	†	†

into non-overlapping sub-tracks of 5 frames and obtain must-link and must-not-link constraints. We follow the same protocols as described in Chapter 3 (Sharma et al., 2019a), Chapter 4 (Sharma et al., 2020) and Chapter 5 (Roethlingshoefer et al., 2019) and mine the same number of positive and negative pairs for training the methods. For testing, we decompose each tracklet into non-overlapping sub-tracks of 5 frames, we then extract feature maps for each sub-track followed by taking average over all the groups of a track to make a track-level representation which is then fed to HAC clustering with fixed number of clusters. Finally, we evaluate the quality of clustering via Accuracy and B-Cubed Precision (P), Recall (R) and F1-measures (F). We use the same training, testing and evaluation setup as described in Chapter 3 (Sharma et al., 2019a), Chapter 4 (Sharma et al., 2020) and Chapter 5 (Roethlingshoefer et al., 2019) for a fair comparison.

Results. We report clustering performance on training videos in Table 6.12 and Table 6.13. Note that all our models are trained in an unsupervised manner, or with automatically generated labels. The goal of this experiment is to show that TCBP encoding can improve the base performance.

In addition for ACCIO, as in (Zhang et al., 2016b), Table 6.14 (num. clusters = 40) shows that our encoding methods improve performance by a significant margin over the state-of-the-art.

Table 6.13: Performance comparison of different methods when integrated with TCBP on ACCIO with 36 clusters. † indicates OUT_OF_MEMORY.

Methods	#cluster=36 P	R	F
JFAC (ECCV '16) (Zhang et al., 2016b)	0.690	0.350	0.460
TSiam (FG '19) (Sharma et al., 2019a)	0.749	0.382	0.506
SSiam (FG '19) (Sharma et al., 2019a)	0.766	0.386	0.514
CCL (FG '20) (Sharma et al., 2020)	0.779	0.402	0.530
FGG (ACMMM '19) (Roethlingshoefer et al., 2019)	0.654	**0.728**	**0.689**
with TCBP			
TSiam	0.767	0.401	0.527
SSiam	0.773	0.408	0.534
CCL	**0.792**	**0.415**	**0.544**
FGG	†	†	†

On a practical side, for FGG (see Chapter 5) training, we found that for all datasets when we integrated FGG with TCBP resulted in out-of-memory issue. The reason behind this is at this moment the whole graph of FGG needs to be present in the memory that means for large scale datasets it causes run time limitations. FGG when integrated with TCBP results to be much more computationally and memory-wise expensive, thus leading to out-of-memory.

6.5 Summary

In this chapter, we introduced a novel task of ordering scrambled video clips in time, and presented an approach that leverages multimodal semantic concepts. We proposed a network architecture to learn a compact multimodal video clip representation that jointly encodes images, audio, video, and text and learns parameters in a self-supervised manner. In particular, Temporal Compact Bilinear Pooling extended CBP to effectively encode multiple temporal segments of a video clip. We evaluated our methods on temporal ordering of video clips and a multimodal video clip retrieval task, where, multiple modalities and TCBP proved to be helpful. We also demonstrated that TCBP is a strong encoding method that performs well on other video tasks such as action recognition and video face clustering. We strongly believe that complete understanding of video clips can only be achieved by analyzing all modalities jointly,

Table 6.14: Performance comparison of different methods when integrated with TCBP on ACCIO with 40 clusters. † indicates OUT_OF_MEMORY.

Methods	#clusters=40 P	R	F
DIFFRAC-DeepID2$^+$ (ICCV '11) (Zhang et al., 2016b)	0.557	0.213	0.301
WBSLRR-DeepID2$^+$ (ECCV '14) (Zhang et al., 2016b)	0.502	0.206	0.292
HMRF-DeepID2$^+$ (CVPR '13) (Zhang et al., 2016b)	0.599	0.230	0.332
JFAC (ECCV '16) (Zhang et al., 2016b)	0.711	0.352	0.471
TSiam (FG '19) (Sharma et al., 2019a)	0.763	0.362	0.491
SSiam (FG '19) (Sharma et al., 2019a)	0.777	0.371	0.502
CCL (FG '20) (Sharma et al., 2020)	0.786	0.392	0.523
FGG (ACMMM '19) (Roethlingshoefer et al., 2019)	0.663	**0.724**	**0.692**
with TCBP			
TSiam	0.772	0.371	0.501
SSiam	0.786	0.379	0.511
CCL	**0.795**	**0.401**	**0.533**
FGG	†	†	†

and hope that such multimodal representations will inspire the community in the future.

Chapter 7

Summary and future work

In this thesis, we have focused on self-supervised face representation learning, wherein we proposed methods to automatically generate pseudo-labels for training a neural network. Specifically, we have shown that with our proposed new techniques to generate weak-labels based on sorting distances (*i.e.* ranking), clustering algorithm and video constraints, one can efficiently learn discriminative face representations, and thus improve video face clustering. We now conclude this thesis by summarising the contributions and conclusions of each chapter. We also discuss promising research directions for future research.

7.1 Ranking-based learning (Chapter 3)

In Chapter 3, we propose two variants of *discriminative* approaches that build upon deep network representations to learn facial representations: Track-supervised Siamese network (TSiam) and Self-supervised Siamese network (SSiam). In Track-supervised Siamese Network (TSiam), we include additional negative training pairs for singleton tracks – tracks that are not temporally co-occurring with any others in contrast to previous methods *e.g.* Cinbis et al. (2011). In Self-supervised Siamese Network (SSiam), we leverage dynamic generation of positive and negative pairs based on sorting distances (*i.e.* ranking) on a subset of frames and do not have to only rely on track-level information that is typically used. Note that the methods proposed in this paper are either fully unsupervised, or use supervision that is obtained automatically, hence can be thought as unsupervised.

Future work. In terms of future work, there are several interesting extensions to consider: in the current setup in SSiam the positive and negative pairs are obtained by sorting distances (*i.e.* ranking) on a subset of samples as a pre-processing step - we believe one can easily integrate SSiam as a loss function(*online learning*) where mining of samples can be jointly optimized along with the representation learning objective. Further, since, SSiam can mine positive and negative pairs without the need for tracking, thus additionally enabling its applicability to image collections, such as photo albums.

7.2 CLUSTERING-BASED LEARNING (CHAPTER 4)

In Chapter 4, we propose to improve face representations using positive and negative pairs obtained through clustering and video level constraints. The main contribution of our work is to utilize automatically discovered partitions obtained from a clustering algorithm as weak supervision for learning improved face representations. In particular, we propose a new clustering-based representation learning approach that uses labels obtained from clustering along with inherent video constraints to learn discriminative face features. As annotating datasets is costly and difficult, using label-free and weak supervision obtained from a clustering algorithm as a proxy learning task is promising, and we clearly show that this helps improve the performance for video face clustering.

Our work aims at improving face representations utilizing both weak cluster labels along with video constraints. In particular, our method uses FINCH-clustering algorithm (Sarfraz et al., 2019) that creates several partitions of the data *without needing to know the number of clusters*. Early partitions of FINCH often produce many more clusters than the true number of categories, yielding *small clusters of very high purity*. Through our analysis we show that creating positive and negative training pairs using high purity partitions outperforms pairs obtained from partitions closer to the true number of characters.

In summary, we show how pseudo-labels from clustering can be used effectively: (i) without the need to know the number of clusters, and (ii) by creating positive/negative pairs at a high-purity stage of clustering.

Future work. Using FINCH cluster labels as weak-labels results in high purity is indeed an important aspect. Though obtaining negative pairs from farthest clusters makes it difficult to generate hard samples. In future work, it is worth investigating directions for negative pairs generation, rather than using the farthest clusters.

Also, while previous methods like TSiam obtain positive pairs from faces within a track, CCL creates positive pairs within FINCH clusters. In our analysis, we find that FINCH clusters even at the high-purity setting often contain faces from more than one track. This means that our learning approach sees relatively hard positive pairs, and standard (neither hard nor easy) negative pairs. We believe that jointly optimizing the clustering loss together with the contrastive loss is a good future direction to learn representations.

7.3 Graph-based learning (Chapter 5)

In Chapter 5 we present face grouping on graphs (FGG), a method for unsupervised fine-tuning of deep face feature representations. By enabling feature representations to update their representations in relation to other representations instead of globally. The representations of face-tracks are related using a graph structure that is induced over temporal and similarity-based constraints, where each node represents a face sub-track and edges indicate either that two tracks belong to the same, or a different person. Using graph neural networks, the features communicate over the edges allowing each track's feature to exchange information with its neighbors. We use separate learnable weight tensors: a *must-link* tensor and a *cannot-link* tensor per edge type and a layer to learn the representation that is aware of the manifold on which the representations of each character live. Further, to increase the number of samples for each character while obtaining a robust estimation of the manifold, we split each face-track into partial tracks (sub-tracks) and pool the feature representations over each sub-track.

Future work. In future work, one could interpret the approach as a link prediction problem, in which one wants to learn whether (and what kind of) edge exists between two nodes given a graph structure. Another potential future step is a wiser utilization of edges by introducing the signed GCN, where the sign refers to the existence of positive and negative edges, in direction similar to that of the signed graphs (Derr et al., 2018; Kumar et al., 2016; Yuan et al., 2017).

On a practical side, at this moment the whole graph needs to be present in the memory that means for large scale datasets it will cause run time limitations. In order to tackle this issue, one could optimize the FGG using sparse tensor implementation.

7.4 COMPACT REPRESENTATION-BASED LEARNING (CHAPTER 6)

In Chapter 6 we propose Temporal Compact Bilinear Pooling (TCBP) an extension of the Tensor Sketch projection algorithm (Pham and Pagh, 2013) to effectively encode the temporal data (*e.g.* face tracks) in videos into a compact descriptor. TCBP is a form of temporal encoding of several chunks of features into a compact representation. TCBP possesses the ability to capture interactions between each element of the feature representation with one-another over a long-range temporal context. We demonstrated that TCBP is a strong encoding method that performs well on face representation learning, multimodal video clip representation that jointly encodes images, audio, video, and text, and video classification.

Future work. The contributions we have introduced are applicable to other tasks where compact and efficient representation is required. Carrying forward the compact encoding of the chapter and combining it with the video-captioning methods and jointly training is also an interesting extensions to consider.

Short CV

Vivek Sharma

Contact	Sharma.Vivek@live.in
Website	https://vivoutlaw.github.io/

Education and Experience

since 2019	Non-Employee Ph.D. Student / Research Affiliate Prof. Rajiv Gupta, Massachusetts General Hospital Prof. Mauricio Santillana, Boston Children's Hospital Harvard Medical School, Harvard University, USA
since 2019	Research Affiliate Prof. Ramesh Raskar, Camera Culture Group MIT Media Lab, Massachusetts Institute of Technology, USA
since 2017	Research Assistant and Ph.D. Student Computer Vision for Human-Computer Interaction Lab Karlsruhe Institute of Technology, Germany
2015 - 2017	Researcher Prof. Luc Van Gool, VISICS / Computer Vision Lab KU Leuven / ETH Zürich
2012 - 2014	Master of Science (M.Sc.) Informatics and Media Technology, Erasmus Mundus CIMET UJM France, UGR Spain, NTNU Norway, and KIT Germany
2007 - 2011	Bachelor of Technology (B.Tech.) Computer Science & Engineering B.K. Birla Institute of Engineering and Technology, Pilani, India

Publications

Vivek Sharma, M Saquib Sarfraz, and Rainer Stiefelhagen. A simple and effective technique for face clustering in tv series. In *Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR) workshop on Brave New Motion Representations*, 2017a.

Vivek Sharma, Ali Diba, Davy Neven, Michael S Brown, Luc Van Gool, and Rainer Stiefelhagen. Classification-driven dynamic image enhancement. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

Congcong Wang*, Vivek Sharma*, Yu Fan, Faouzi Alaya Cheikh, Azeddine Beghdadi, Ole Jacob Elle, and Rainer Stiefelhagen. Can image enhancement be beneficial to find smoke images in laparoscopic surgery? In *Proceedings of the Color and Imaging Conference (CIC)*, 2018.

Vivek Sharma, Makarand Tapaswi, M Saquib Sarfraz, and Rainer Stiefelhagen. Self-supervised learning of face representations for video face clustering. In *Proceedings of the IEEE International Conference on Automatic Face and Gesture Recognition (FG)*, 2019a.

Vivek Sharma, Makarand Tapaswi, M Saquib Sarfraz, and Rainer Stiefelhagen. Video face clustering with self-supervised representation learning. *IEEE Transactions on Biometrics, Behavior, and Identity Science (T-BIOM)*, 2019b.

Veith Röthlingshöfer*, Vivek Sharma*, and Rainer Stiefelhagen. Self-supervised face-grouping on graphs. In *Proceedings of the ACM International Conference on Multimedia (ACMMM)*, 2019.

Vivek Sharma, Makarand Tapaswi, and Rainer Stiefelhagen. Deep multimodal feature encoding for video ordering. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV) workshop on Large Scale Holistic Video Understanding*, 2019c.

M Saquib Sarfraz, Vivek Sharma, and Rainer Stiefelhagen. Efficient parameter-free clustering using first neighbor relations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

Ali Diba*, Vivek Sharma*, Rainer Stiefelhagen, and Luc Van Gool. Weakly supervised object discovery by generative adversarial & ranking networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) workshop on Efficient Feature Representation and Learning in Computer Vision*, 2019a.

Ali Diba*, Vivek Sharma*, Luc Van Gool, and Rainer Stiefelhagen. Dynamonet: Dynamic action and motion network. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2019b.

Vivek Sharma, Makarand Tapaswi, Saquib Sarfraz, and Rainer Stiefelhagen. Clustering based contrastive learning for improving face representations. In *Proceedings of the IEEE International Conference on Automatic Face and Gesture Recognition (FG)*, 2020a.

Ali Diba*, Mohsen Fayyaz*, Vivek Sharma*, Manohar Paluri, Jurgen Gall, Rainer Stiefelhagen, and Luc Van Gool. Large scale holistic video understanding. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

Vivek Sharma, Gabriela Csurka, Naila Murray, Diane Larlus, M Saquib Sarfraz, and Rainer Stiefelhagen. Unsupervised meta-domain adaptation for fashion retrieval. *Under Review*, 2020b.

Ali Diba*, Vivek Sharma*, and Luc Van Gool. Deep temporal linear encoding networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Ali Diba, Vivek Sharma, Ali Pazandeh, Hamed Pirsiavash, and Luc Van Gool. Weakly supervised cascaded convolutional networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Vivek Sharma, Jon Yngve Hardeberg, and Sony George. Rgb-nir image enhancement by fusing bilateral and weighted least squares filters. *Journal of Imaging Science and Technology*, 2017b.

Ali Diba, Mohsen Fayyaz, Vivek Sharma, A Hossein Karami, M Mahdi Arzani, Rahman Yousefzadeh, and Luc Van Gool. Temporal 3d convnets using temporal transition layer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) workshop on Brave New Ideas for Video Understanding*, 2018a.

Ali Diba, Mohsen Fayyaz, Vivek Sharma, M Mahdi Arzani, Rahman Yousefzadeh, Juergen Gall, and Luc Van Gool. Spatio-temporal channel correlation networks for action classification. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018b.

Vivek Sharma, Praneeth Vepakomma, Tristan Swedish, Ken Chang, Jayashree Kalpathy-Cramer, and Ramesh Raskar. Expertmatcher: Automating ml model selection for users in resource constrained countries. In *Proceedings of the Neural Information Processing Systems (NeurIPS) workshop on Machine learning for the Developing World*, 2019d.

Vivek Sharma, Praneeth Vepakomma, Tristan Swedish, Ken Chang, Jayashree Kalpathy-Cramer, and Ramesh Raskar. Expertmatcher: Automating ml model selection for clients using hidden representations. In *Proceedings of the Neural Information Processing Systems (NeurIPS) workshop on Robust AI in Financial Services: Data, Fairness, Explainability, Trustworthiness, and Privacy*, 2019e.

Bibliography

H. Abdi and L. J. Williams. Principal component analysis. *WIREs Computational Statistics*, 2010. xvii, 93, 95

R. Aljundi, P. Chakravarty, and T. Tuytelaars. Who's that Actor? Automatic Labelling of Actors in TV series starting from IMDB Images. In *Asian Conference on Computer Vision (ACCV)*, 2016. 28

E. Amigó, J. Gonzalo, J. Artiles, and F. Verdejo. A comparison of extrinsic clustering evaluation metrics based on formal constraints. *Information retrieval*, 2009. 24, 25

R. Arandjelović, P. Gronat, A. Torii, T. Pajdla, and J. Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 102, 114

R. Arandjelovic and A. Zisserman. Look, listen and learn. In *International Conference on Computer Vision (ICCV)*, 2017. 102

R. Arandjelović and A. Zisserman. Objects that sound. In *European Conference on Computer Vision (ECCV)*, 2018. 102

Y. Aytar, C. Vondrick, and A. Torralba. Soundnet: Learning sound representations from unlabeled video. In *Workshop at Advances in Neural Information Processing Systems (NIPSW)*, 2016. 101, 102, 109

M. Azab, N. Kojima, J. Deng, and R. Mihalcea. Representing movie characters in dialogues. In *Computational Natural Language Learning (CoNLL)*, 2019. 102

A. Bagga and B. Baldwin. Entity-based cross-document coreferencing using the vector space model. In *Association of Computational Linguistics (ACL)*, 1998. 24

T. Basha, Y. Moses, and S. Avidan. Photo sequencing. In *European Conference on Computer Vision (ECCV)*, 2012. 98, 101

P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv:1806.01261*, 2018. 75, 76, 77

M. Bäuml, M. Tapaswi, and R. Stiefelhagen. Semi-supervised Learning with Constraints for Person Identification in Multimedia Data. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013. xix, 2, 20, 21, 22, 28, 32, 49, 74, 75, 76

M. Berman, A. Rannen Triki, and M. B. Blaschko. The lovász-softmax loss: a tractable surrogate for the optimization of the intersection-over-union measure in neural networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 13

X. Bresson and T. Laurent. Residual gated graph convnets. *arXiv:1711.07553*, 2017. 76, 77, 82

K. Briggman, W. Denk, S. Seung, M. N. Helmstaedter, and S. C. Turaga. Maximin affinity learning of image segmentation. In *Workshop at Advances in Neural Information Processing Systems (NIPSW)*. 13

M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 2017. 76

C. J. C. Burges, B. Schölkopf, and A. J. Smola. *Advances in Kernel Methods: Support Vector Learning*. MIT Press, 1998. 104

D. Cai, X. He, and J. Han. Locally consistent concept factorization for document clustering. *IEEE Transactions on Knowledge and Data Engineering*, 2011. 77

Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman. VGGFace2: A Dataset for Recognising Faces across Pose and Age. In *International Conference on Automatic Face and Gesture Recognition (FG)*, 2018. xxi, 1, 2, 19, 29, 35, 41, 42, 60, 64, 74, 77, 85, 86, 119

M. Caron, P. Bojanowski, A. Joulin, and M. Douze. Deep clustering for unsupervised learning of visual features. In *European Conference on Computer Vision (ECCV)*, 2018. 56, 59

J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 98, 99, 109

P. Cascante-Bonilla, K. Sitaraman, M. Luo, and V. Ordonez. Moviescope: Large-scale analysis of movies using multiple modalities. *arXiv:1908.03180*, 2019. 99

S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005. 13

A. R. Chowdhury, T.-Y. Lin, S. Maji, and E. Learned-Miller. One-to-many face recognition with bilinear cnns. 2016. 102, 104

R. G. Cinbis, J. Verbeek, and C. Schmid. Unsupervised Metric Learning for Face Identification in TV Video. In *International Conference on Computer Vision (ICCV)*, 2011. 20, 21, 22, 28, 30, 32, 37, 48, 49, 63, 70, 92, 123

D. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). In *International Conference on Learning Representations (ICLR)*, 2016. 16, 84

T. Cour, B. Sapp, A. Nagle, and B. Taskar. Talking pictures: Temporal grouping and dialog-supervised person recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010. 78

G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *Workshop at Conference on European Conference on Computer Vision (ECCVW)*, 2004. 102

S. Datta, G. Sharma, and C. Jawahar. Unsupervised learning of face representations. In *International Conference on Automatic Face and Gesture Recognition (FG)*, 2018. 28, 33, 37, 43, 58, 63

B. De Brabandere, D. Neven, and L. Van Gool. Semantic instance segmentation with a discriminative loss function. In *Workshop at Conference on Computer Vision and Pattern Recognition (CVPRW)*. 13

Z. Deng, R. Navarathna, P. Carr, S. Mandt, Y. Yue, I. Matthews, and G. Mori. Factorized Variational Autoencoders for Modeling Audience Reactions to Movies. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 31

T. Derr, Y. Ma, and J. Tang. Signed graph convolutional networks. In *International Conference on Data Mining (ICDM)*, 2018. 75, 76, 77, 95, 125

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Association of Computational Linguistics (ACL)*, 2018. 56

A. Diba, A. M. Pazandeh, and L. Van Gool. Efficient two-stream motion and appearance 3d cnns for video classification. In *Workshop at Conference on European Conference on Computer Vision (ECCVW)*, 2016. 102

A. Diba, V. Sharma, and L. Van Gool. Deep temporal linear encoding networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 29, 98, 102, 104, 116, 118, 119

A. Diba, M. Fayyaz, V. Sharma, A. Hossein Karami, M. Mahdi Arzani, R. Yousefzadeh, and L. Van Gool. Temporal 3d convnets using temporal transition layer. In *Workshop at Conference on Computer Vision and Pattern Recognition (CVPRW)*, 2018a. 60, 98

A. Diba, V. Sharma, L. Van Gool, and R. Stiefelhagen. Dynamonet: Dynamic action and motion network. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019b. 98

C. Dicle, B. Yilmaz, O. Camps, and M. Sznaier. Solving temporal puzzles. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 98, 101

N. Dilokthanakul, P. A. Mediano, M. Garnelo, M. C. Lee, H. Salimbeni, K. Arulkumaran, and M. Shanahan. Deep Unsupervised Clustering with Gaussian Mixture Variational Autoencoders. *arXiv:1611.02648*, 2016. 32

C. Doersch. Tutorial on Variational Autoencoders. *arXiv:1606.05908*, 2016. 40

C. Doersch, A. Gupta, and A. A. Efros. Unsupervised visual representation learning by context prediction. In *International Conference on Computer Vision (ICCV)*, 2015. 56, 100

J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 118

A. Dosovitskiy, J. T. Springenberg, M. Riedmiller, and T. Brox. Discriminative unsupervised feature learning with convolutional neural networks. In *Workshop at Advances in Neural Information Processing Systems (NIPSW)*, 2014. 100

H. Doughty, W. Mayol-Cuevas, and D. Damen. The pros and cons: Rank-aware temporal attention for skill determination in long videos. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 108

J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research (JMLR)*, 2011. 15

A. Ephrat, I. Mosseri, O. Lang, T. Dekel, K. Wilson, A. Hassidim, W. T. Freeman, and M. Rubinstein. Looking to listen at the cocktail party: A speaker-independent audio-visual model for speech separation. In *ACM SIGGRAPH*, 2018. 102

M. Everingham, J. Sivic, and A. Zisserman. "Hello! My name is ... Buffy" – Automatic Naming of Characters in TV Video. In *British Machine Vision Conference (BMVC)*, 2006. 28, 32, 48, 49

B. Fernando, H. Bilen, E. Gavves, and S. Gould. Self-supervised video representation learning with odd-one-out networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 33, 56, 99, 100, 101

A. Fukui, D. H. Park, D. Yang, A. Rohrbach, T. Darrell, and M. Rohrbach. Multimodal compact bilinear pooling for visual question answering and visual grounding. In *Association of Computational Linguistics (ACL)*, 2016. 99, 104

Y. Gao, O. Beijbom, N. Zhang, and T. Darrell. Compact bilinear pooling. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 99, 104, 105

E. Ghaleb, M. Tapaswi, Z. Al-Halah, H. K. Ekenel, and R. Stiefelhagen. Accio: A Dataset for Face Track Retrieval in Movies Across Age. In *International Conference on Multimedia Retrieval (ICMR)*, 2015. 20, 22, 76

J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. Neural message passing for quantum chemistry. In *International Conference on Machine Learning (ICML)*, 2017. 75, 76, 77

R. Girdhar, D. Ramanan, A. Gupta, J. Sivic, and B. Russell. Actionvlad: Learning spatio-temporal aggregation for action classification. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 102, 114, 119

X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011. 10, 16

Y. Gong, L. Wang, R. Guo, and S. Lazebnik. Multi-scale orderless pooling of deep convolutional activation features. In *European Conference on Computer Vision (ECCV)*, 2014. 102

I. Goodfellow, Y. Bengio, and A. Courville. Deep learning. book in preparation for mit press. *URL¡ http://www. deeplearningbook. org*, 2016. 10

M. Guillaumin, J. Verbeek, and C. Schmid. Is that you? Metric Learning Approaches for Face Identification. In *International Conference on Computer Vision (ICCV)*, 2009. 28

W. Guo, H. Huang, X. Kong, and R. He. Learning disentangled representation for cross-modal retrieval with deep mutual information estimation. In *ACM Multimedia (MM)*, 2019. 99

X. Guo, L. Gao, X. Liu, and J. Yin. Improved deep embedded clustering with local structure preservation. In *International Joint Conferences on Artificial Intelligence (IJCAI)*, 2017. 56, 59

Y. Guo, L. Zhang, Y. Hu, X. He, and J. Gao. MS-Celeb-1M: A Dataset and Benchmark for Large-Scale Face Recognition. In *European Conference on Computer Vision (ECCV)*, 2016. 41, 64, 119

S. Gupta, J. Hoffman, and J. Malik. Cross modal distillation for supervision transfer. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 102

R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality Reduction by Learning an Invariant Mapping. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006. 12, 36, 58, 63, 84

W. L. Hamilton, R. Ying, and J. Leskovec. Representation learning on graphs: Methods and applications. *IEEE Data Engineering Bulletin*, 2017. 77

K. Hara, H. Kataoka, and Y. Satoh. Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet? In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 98, 109, 118

M.-L. Haurilet, M. Tapaswi, Z. Al-Halah, and R. Stiefelhagen. Naming TV Characters by Watching and Analyzing Dialogs. 2016. 32

K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *International Conference on Computer Vision (ICCV)*, 2015. 16, 82

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 17, 18, 56, 77, 82, 99, 101

X. He, Y. Peng, and L. Xie. A new benchmark and approach for fine-grained cross-media retrieval. In *ACM Multimedia (MM)*, 2019. 99

Y. He, K. Cao, C. Li, and C. C. Loy. Merge or not? learning to group faces via imitation learning. In *Association for the Advancement of Artificial Intelligence (AAAI)*, 2018. 33, 58, 92, 93

G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv:1503.02531*, 2015. 102

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 1997. 99

C. Hori, T. Hori, T.-Y. Lee, Z. Zhang, B. Harsham, J. R. Hershey, T. K. Marks, and K. Sumi. Attention-based multimodal fusion for video description. In *International Conference on Computer Vision (ICCV)*, 2017. 99, 102

X. Hou, L. Shen, K. Sun, and G. Qiu. Deep Feature Consistent Variational Autoencoder. 2017. 31

P. Hu and D. Ramanan. Finding Tiny Faces. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 29

G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller. Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments. In *Workshop at Conference on European Conference on Computer Vision (ECCVW)*, 2008. 29

G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 17, 18

Q. Huang, W. Liu, and D. Lin. Person search in videos with one portrait through visual and temporal links. In *European Conference on Computer Vision (ECCV)*, 2018. 76

P. J. Huber. Robust estimation of a location parameter. In *Breakthroughs in statistics*. 1992. 11

S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, 2015. 84

H. Ishfaq, A. Hoogi, and D. Rubin. TVAE: Deep Metric Learning Approach for Variational Autoencoder. In *Workshop at International Conference on Learning Representations (ICLRW)*, 2018. 32

D. Jayaraman and K. Grauman. Slow and steady feature analysis: higher order temporal coherence in video. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 101

Z. Jiang, Y. Zheng, H. Tan, B. Tang, and H. Zhou. Variational deep embedding: An unsupervised and generative approach to clustering. In *International Joint Conferences on Artificial Intelligence (IJCAI)*, 2017. 56, 59

Q. Jin, J. Chen, S. Chen, Y. Xiong, and A. Hauptmann. Describing videos using multi-modal fusion. In *ACM Multimedia (MM)*, 2016. 99

S. Jin, H. Su, C. Stauffer, and E. Learned-Miller. End-to-end Face Detection and Cast Grouping in Movies using Erdős–Rényi Clustering. In *International Conference on Computer Vision (ICCV)*, 2017. 28, 32, 58

A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. 116, 118

T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive Growing of GANs for Improved Quality, Stability, and Variation. In *International Conference on Learning Representations (ICLR)*, 2018. 31

A. Kendall and R. Cipolla. Geometric loss functions for camera pose regression with deep learning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 13

G. Kim, L. Sigal, and E. P. Xing. Joint summarization of large-scale collections of web images and videos for storyline reconstruction. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. 99

D. Kingma and M. Welling. Auto-encoding Variational Bayes. In *International Conference on Learning Representations (ICLR)*, 2014. 30, 39, 40

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. 2015. 15, 84

T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. 2017. 75, 76, 77, 78, 80, 82

R. Kiros, Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, and S. Fidler. Skip-thought vectors. In *Workshop at Advances in Neural Information Processing Systems (NIPSW)*, 2015. 99

G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter. Self-normalizing neural networks. In *Workshop at Advances in Neural Information Processing Systems (NIPSW)*, 2017. 16

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Workshop at Advances in Neural Information Processing Systems (NIPSW)*, 2012. 17, 18

H. Kuehne, H. Jhuang, R. Stiefelhagen, and T. Serre. Hmdb51: A large video database for human motion recognition. In *High Performance Computing in Science and Engineering*. 2013. 116

S. Kumar, F. Spezzano, V. S. Subrahmanian, and C. Faloutsos. Edge weight prediction in weighted signed networks. In *Conference on International Conference on Data Mining (ICDM)*, 2016. 95, 125

B. Laxton, J. Lim, and D. Kriegman. Leveraging temporal, contextual and ordering constraints for recognizing complex activities in video. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007. 101

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based Learning Applied to Document Recognition. *Proceedings of IEEE*, 1998. 31

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1989. 16, 17

Y. Li, D. Tarlow, M. Brockschmidt, and R. S. Zemel. Gated graph sequence neural networks. In *International Conference on Learning Representations (ICLR)*, 2016. 76, 77, 78, 82

M. Limmer and H. P. Lensch. Infrared colorization using deep convolutional neural networks. In *IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2016. 102

T.-Y. Lin, A. RoyChowdhury, and S. Maji. Bilinear cnn models for fine-grained visual recognition. In *International Conference on Computer Vision (ICCV)*, 2015. 102, 104

A. B. Lindbo Larsen, S. K. Sønderby, H. Larochelle, and O. Winther. Autoencoding beyond pixels using a learned similarity metric. *arXiv:1512.09300*, 2015. 31

Y. Liu, J. Yan, and W. Ouyang. Quality Aware Network for Set to Set Recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 60

A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *International Conference on Machine Learning (ICML)*, 2013. 16

T. Maharaj, N. Ballas, A. Rohrbach, A. C. Courville, and C. J. Pal. A dataset and exploration of models for understanding video data through fill-in-the-blank question-answering. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 110

K. Matzen and N. Snavely. Scene chronology. In *European Conference on Computer Vision (ECCV)*, 2014. 98, 101

A. Miech, J.-B. Alayrac, P. Bojanowski, I. Laptev, and J. Sivic. Learning from video and text via large-scale discriminative clustering. In *International Conference on Computer Vision (ICCV)*, 2017a. 32

A. Miech, I. Laptev, and J. Sivic. Learnable pooling with context gating for video classification. In *Workshop at Conference on Computer Vision and Pattern Recognition (CVPRW)*, 2017b. 102, 114

T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Workshop at Advances in Neural Information Processing Systems (NIPSW)*, 2013. 56

I. Misra, C. L. Zitnick, and M. Hebert. Shuffle and learn: unsupervised learning using temporal order verification. In *European Conference on Computer Vision (ECCV)*, 2016. 33, 56, 99, 100, 101

J. G. Moreno and G. Dias. Adapted b-cubed metrics to unbalanced datasets. In *ACM Conference on Research and Development in Information Retrieval*, 2015. 24

Y. Moses, S. Avidan, et al. Space-time tradeoffs in photo sequencing. In *International Conference on Computer Vision (ICCV)*, 2013. 98, 101

A. Nagrani and A. Zisserman. From Benedict Cumberbatch to Sherlock Holmes: Character Identification in TV series without a Script. In *British Machine Vision Conference (BMVC)*, 2017. 28, 32

A. Nagrani, S. Albanie, and A. Zisserman. Learnable pins: Cross-modal embeddings for person identity. In *European Conference on Computer Vision (ECCV)*, 2018a. 102

A. Nagrani, S. Albanie, and A. Zisserman. Seeing voices and hearing faces: Cross-modal biometric matching. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018b. 102

A. Narayanan, M. Chandramohan, R. Venkatesan, L. Chen, Y. Liu, and S. Jaiswal. graph2vec: Learning distributed representations of graphs. In *International Workshop on Mining and Learning with Graphs (MLG)*, 2017. 77

J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng. Multimodal deep learning. In *International Conference on Machine Learning (ICML)*, 2011. 102

A. Owens and A. A. Efros. Audio-visual scene analysis with self-supervised multisensory features. In *European Conference on Computer Vision (ECCV)*, 2018. 102

A. Owens, P. Isola, J. McDermott, A. Torralba, E. H. Adelson, and W. T. Freeman. Visually indicated sounds. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016a. 102

A. Owens, J. Wu, J. H. McDermott, W. T. Freeman, and A. Torralba. Ambient sound provides supervision for visual learning. In *European Conference on Computer Vision (ECCV)*, 2016b. 102

S. Parekh, S. Essid, A. Ozerov, N. Q. Duong, P. Pérez, and G. Richard. Weakly supervised representation learning for unsynchronized audio-visual events. *Journal of Multimedia*, 2018. 102

O. M. Parkhi, K. Simonyan, A. Vedaldi, and A. Zisserman. A Compact and Discriminative Face Track Descriptor. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. 32, 60

O. M. Parkhi, A. Vedaldi, and A. Zisserman. Deep Face Recognition. In *British Machine Vision Conference (BMVC)*, 2015. 1, 2, 19, 29, 42, 56

A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017. 85

G. Paul, K. Elie, M. Sylvain, O. Jean-Marc, and D. Paul. A conditional random field approach for audio-visual people diarization. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014. 32

Y. Peng, J. Qi, X. Huang, and Y. Yuan. Ccl: Cross-modal correlation learning with multigrained fusion by hierarchical network. *IEEE Transactions on Multimedia*, 2018. 99

J. Pennington, R. Socher, and C. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2014. 109

F. Perronnin, J. Sánchez, and T. Mensink. Improving the fisher kernel for large-scale image classification. In *European Conference on Computer Vision (ECCV)*, 2010. 102

N. Pham and R. Pagh. Fast and scalable polynomial kernels via explicit feature maps. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2013. 6, 99, 102, 105, 126

L. C. Pickup, Z. Pan, D. Wei, Y. Shih, C. Zhang, A. Zisserman, B. Scholkopf, and W. T. Freeman. Seeing the Arrow of Time. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. 101

F. Qi, X. Yang, and C. Xu. A unified framework for multimodal domain adaptation. In *ACM Multimedia (MM)*, 2018. 99

N. Qian. On the momentum term in gradient descent learning algorithms. *Neural networks*, 1999. 15

P. Ramachandran, B. Zoph, and Q. V. Le. Searching for activation functions. *arXiv:1710.05941*, 2017. 16

V. Ramanathan, A. Joulin, P. Liang, and L. Fei-Fei. Linking people in videos with "their" names using coreference resolution. In *European Conference on Computer Vision (ECCV)*, 2014. 28

L. Ren, S. Liu, H. Huang, J. Han, S. Yan, and B. Li. Finding images by dialoguing with image. In *ACM Multimedia (MM)*, 2019. 99

V. Roethlingshoefer, V. Sharma, and R. Stiefelhagen. Self-supervised face-grouping on graph. In *ACM Multimedia (MM)*, 2019. 33, 100, 120, 121, 122

A. Rohrbach, M. Rohrbach, S. Tang, S. J. Oh, and B. Schiele. Generating Descriptions with Grounded and Co-Referenced People. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017a. 28

A. Rohrbach, A. Torabi, M. Rohrbach, N. Tandon, C. Pal, H. Larochelle, A. Courville, and B. Schiele. Movie description. *International Journal of Computer Vision (IJCV)*, 2017b. 28, 110

M. Roth, M. Bäuml, R. Nevatia, and R. Stiefelhagen. Robust Multi-pose Face Tracking by Multi-stage Tracklet Association. In *International Conference on Pattern Recognition (ICPR)*, 2012. 48, 49

S. Ruder. An overview of gradient descent optimization algorithms. *arXiv:1609.04747*, 2016. xv, 15

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985. 10, 15

O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 2015. 18, 109

F. Sadeghi, J. R. Tena, A. Farhadi, and L. Sigal. Learning to select and order vacation photographs. 2015. 98, 101

M. S. Sarfraz and O. Hellwich. An efficient front-end facial pose estimation system for face recognition. *Pattern Recognition and Image Analysis*, 2008a. 32

M. S. Sarfraz and O. Hellwich. Head pose estimation in face recognition across pose scenarios. *VISAPP (1)*, 2008b. 32

M. S. Sarfraz and O. Hellwich. Probabilistic learning for fully automatic face recognition across pose. *Image and Vision Computing*, 2010. 32

M. S. Sarfraz and M. H. Khan. A probabilistic framework for patch based vehicle type recognition. *VISAPP*, 2011. 59

M. S. Sarfraz, A. Schumann, A. Eberle, and R. Stiefelhagen. A pose-sensitive embedding for person re-identification with expanded cross neighborhood re-ranking. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 29

M. S. Sarfraz, V. Sharma, and R. Stiefelhagen. Efficient parameter-free clustering using first neighbor relations. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 33, 57, 58, 59, 61, 62, 70, 71, 92, 93, 94, 111, 124

G. Schindler, F. Dellaert, and S. B. Kang. Inferring temporal order of images from 3d structure. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007. 98, 101

F. Schroff, D. Kalenichenko, and J. Philbin. FaceNet: A Unified Embedding for Face Recognition and Clustering. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 2, 12, 13, 19, 29

D. Sculley. Web-scale K-means Clustering. In *International Conference on World Wide Web*, 2010. 62

V. Sharma and L. Van Gool. Image-level classification in hyperspectral images using feature descriptors, with application to face recognition. *arXiv:1605.03428*, 2016. 32

V. Sharma, A. Diba, T. Tuytelaars, and L. Van Gool. Hyperspectral cnn for image classification & band selection, with application to face recognition. *Technical report KUL/ESAT/PSI/1604, KU Leuven, ESAT, Leuven, Belgium*, 2016. 32

V. Sharma, M. S. Sarfraz, and R. Stiefelhagen. A simple and effective technique for face clustering in tv series. In *Workshop at Conference on Computer Vision and Pattern Recognition (CVPRW)*, 2017. 36, 49, 61, 70, 92

V. Sharma, A. Diba, D. Neven, M. S. Brown, L. Van Gool, and R. Stiefelhagen. Classification driven dynamic image enhancement. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 29

V. Sharma, M. Tapaswi, M. S. Sarfraz, and R. Stiefelhagen. Self-supervised learning of face representations for video face clustering. In *International Conference on Automatic Face and Gesture Recognition (FG)*, 2019a. xx, xxi, xxii, 57, 61, 63, 66, 70, 71, 74, 78, 79, 81, 87, 88, 89, 91, 92, 93, 94, 99, 100, 120, 121, 122

V. Sharma, M. Tapaswi, and R. Stiefelhagen. Deep multimodal feature encoding for video ordering and retrieval tasks. In *International Conference on Computer Vision (ICCV): Workshop on Holistic Video Understanding (HVU)*, 2019b. 56

V. Sharma, M. Tapaswi, and R. Stiefelhagen. Clustering based contrastive learning for improving face representations. In *International Conference on Automatic Face and Gesture Recognition (FG)*, 2020. xxi, xxii, 89, 92, 93, 94, 120, 121, 122

N. Siddharth, B. Paige, J.-W. van de Meent, A. Desmaison, N. D. Goodman, P. Kohli, F. Wood, and P. Torr. Learning Disentangled Representations with Semi-Supervised Deep Generative Models. In *Workshop at Advances in Neural Information Processing Systems (NIPSW)*, 2017. 31

K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *Workshop at Advances in Neural Information Processing Systems (NIPSW)*, 2014a. 118

K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-scale Image Recognition. In *International Conference on Learning Representations (ICLR)*, 2014b. 17, 18

J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. In *International Conference on Computer Vision (ICCV)*, 2003. 102

J. Sivic, M. Everingham, and A. Zisserman. "Who are you?" – Learning person specific classifiers from video. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009. 28

K. Soomro, A. R. Zamir, and M. Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv:1212.0402*, 2012. 116

N. Srivastava, E. Mansimov, and R. Salakhutdinov. Unsupervised learning of video representations using lstms. In *International Conference on Machine Learning (ICML)*, 2015. 118

A. Subramanya and J. Bilmes. Large scale graph transduction. 2009. 76

C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 17, 18

Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. DeepFace: Closing the Gap to Human-Level Performance in Face Verification. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. 2, 19, 29

P. Tang, X. Wang, B. Shi, X. Bai, W. Liu, and Z. Tu. Deep fishernet for object classification. *IEEE Transactions on Neural Networks and Learning Systems*, 2016. 102

Y. Tang, R. Salakhutdinov, and G. Hinton. Robust Boltzmann Machines for Recognition and Denoising. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. 31

M. Tapaswi, M. Bäuml, and R. Stiefelhagen. "Knock! Knock! Who is it?" Probabilistic Person Identification in TV-Series. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. 22, 32, 74, 76, 102

M. Tapaswi, M. Bäuml, and R. Stiefelhagen. StoryGraphs: Visualizing Character Interactions as a Timeline. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014a. 111

M. Tapaswi, O. M. Parkhi, E. Rahtu, E. Sommerlade, R. Stiefelhagen, and A. Zisserman. Total Cluster: A Person Agnostic Clustering Method for Broadcast Videos. In *Indian Conference on Computer Vision, Graphics and Image Processing (ICVGIP)*, 2014b. xix, 24, 32, 35, 37, 47, 63, 74, 78, 100

M. Tapaswi, Y. Zhu, R. Stiefelhagen, A. Torralba, R. Urtasun, and S. Fidler. Movieqa: Understanding stories in movies through question-answering. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 28

M. Tapaswi, M. T. Law, and S. Fidler. Video face clustering with unknown number of clusters. In *International Conference on Computer Vision (ICCV)*, 2019. 28, 29, 33, 51, 58, 61

J. B. Tenenbaum and W. T. Freeman. Separating style and content with bilinear models. *Neural computation*, 2000. 102

A. Torabi, N. Tandon, and L. Sigal. Learning language-visual embedding for movie understanding with natural-language. *arXiv:1609.08124*, 2016. 110

D. L. Tran, R. Walecki, O. Rudovic, S. Eleftheriadis, B. Schuller, and M. Pantic. DeepCoder: Semi-parametric Variational Autoencoders for Automatic Facial Action Coding. In *International Conference on Computer Vision (ICCV)*, 2017. 31

D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *International Conference on Computer Vision (ICCV)*, 2015. 116, 118

L. Tran, X. Yin, and X. Liu. Representation learning by rotating your faces. 2018. 74

L. van der Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research (JMLR)*, 2008. xvii, 93, 96

A. Vedaldi and K. Lenc. Matconvnet: Convolutional neural networks for matlab. In *ACM Multimedia (MM)*, 2015. 50

I. Vendrov, R. Kiros, S. Fidler, and R. Urtasun. Order-embeddings of images and language. In *International Conference on Learning Representations (ICLR)*, 2016. 13, 108

P. Vicol, M. Tapaswi, L. Castrejon, and S. Fidler. Moviegraphs: Towards understanding human-centric situations from videos. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 28, 76

H. Wang and C. Schmid. Action recognition with improved trajectories. In *International Conference on Computer Vision (ICCV)*, 2013. 118

J. Wang, T. Jebara, and S.-F. Chang. Graph transduction via alternating minimization. In *International Conference on Machine Learning (ICML)*, 2008. 76

L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool. Temporal segment networks: towards good practices for deep action recognition. In *European Conference on Computer Vision (ECCV)*, 2016. 118

R. Wang, H. Guo, L. S. Davis, and Q. Dai. Covariance Discriminative Learning: A Natural and Efficient Approach to Image Set Classification. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. 60

X. Wang and A. Gupta. Unsupervised learning of visual representations using videos. In *International Conference on Computer Vision (ICCV)*, 2015. 33, 56, 99, 100

Z. Wang, L. Zheng, Y. Li, and S. Wang. Linkage based face clustering via graph convolution network. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 77

J. H. Ward Jr. Hierarchical Grouping to Optimize an Objective Function. *Journal of the American Statistical Association*, 1963. 35, 61, 78, 84, 85

D. Wei, J. Lim, A. Zisserman, and W. T. Freeman. Learning and using the arrow of time. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 101

L. Wolf, T. Hassner, and I. Maoz. Face Recognition in Unconstrained Videos with Matched Background Similarity. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011. 29

A. Wu and Y. Han. Multi-modal circulant fusion for video-to-language and backward. In *International Joint Conferences on Artificial Intelligence (IJCAI)*, 2018. 99, 102

B. Wu, S. Lyu, B.-G. Hu, and Q. Ji. Simultaneous Clustering and Tracklet Linking for Multi-face Tracking in Videos. In *International Conference on Computer Vision (ICCV)*, 2013a. 20, 22, 28, 32, 49, 70, 74, 76, 92

B. Wu, Y. Zhang, B.-G. Hu, and Q. Ji. Constrained Clustering and its Application to Face Clustering in Videos. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013b. 20, 21, 22, 28, 32, 37, 47, 49, 63, 70, 78, 92

Y. Wu, H. Liu, J. Li, and Y. Fu. Improving face representation learning with center invariant loss. *Image and Vision Computing*, 2018. 74

S. Xiao, M. Tan, and D. Xu. Weighted Block-sparse Low Rank Representation for Face Clustering in Videos. In *European Conference on Computer Vision (ECCV)*, 2014. 21, 22, 32, 49, 70, 92

J. Xie, R. Girshick, and A. Farhadi. Unsupervised Deep Embedding for Clustering Analysis. In *International Conference on Machine Learning (ICML)*, 2016. 31, 56

D. Xu, J. Xiao, Z. Zhao, J. Shao, D. Xie, and Y. Zhuang. Self-supervised spatiotemporal learning via video clip order prediction. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 101

R. Yan, A. Hauptmann, and R. Jin. Multimedia search with pseudo-relevance feedback. In *International Conference on Image and Video Retrieval*, 2003a. 37, 43, 65

R. Yan, A. G. Hauptmann, and R. Jin. Negative pseudo-relevance feedback in content-based video retrieval. In *ACM Multimedia (MM)*, 2003b. xx, 37, 43, 65, 66

X. Yan, J. Yang, K. Sohn, and H. Lee. Attribute2Image: Conditional Image Generation from Visual Attributes. In *European Conference on Computer Vision (ECCV)*, 2016. 31

J. Yang, K. Yu, Y. Gong, and T. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009. 102

J. Yang, D. Parikh, and D. Batra. Joint unsupervised learning of deep representations and image clusters. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 56, 59

J. Yang, P. Ren, D. Zhang, D. Chen, F. Wen, H. Li, and G. Hua. Neural Aggregation Network for Video Face Recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017a. 60

Z. Yang, Y. Xu, H. Wang, B. Wang, and Y. Han. Multirate multimodal video captioning. In *ACM Multimedia (MM)*, 2017b. 99

Z. Yu, J. Yu, J. Fan, and D. Tao. Multi-modal factorized bilinear pooling with co-attention learning for visual question answering. In *International Conference on Computer Vision (ICCV)*, 2017. 99

S. Yuan, X. Wu, and Y. Xiang. Sne: Signed network embedding. In J. Kim, K. Shim, L. Cao, J.-G. Lee, X. Lin, and Y.-S. Moon (eds), *Advances in Knowledge Discovery and Data Mining*, 2017. 95, 125

B. P. Yuhas, M. H. Goldstein, and T. J. Sejnowski. Integration of acoustic and visual speech signals using neural networks. *IEEE Communications Magazine*, 1989. 102

A. T. Yusuf Aytar, Carl Vondrick. See, hear, and read: Deep aligned representations. In *arXiv:1706.00932*, 2017. 102

M. D. Zeiler. Adadelta: an adaptive learning rate method. *arXiv:1212.5701*, 2012. 15

L. Zhang, D. V. Kalashnikov, and S. Mehrotra. A unified framework for context assisted face clustering. In *International Conference on Multimedia Retrieval (ICMR)*, 2013. 32

S. Zhang, Y. Gong, and J. Wang. Deep metric learning with improved triplet loss for face clustering in videos. In *Pacific Rim Conference on Multimedia*, 2016a. 2, 13, 20, 21, 22, 28, 29, 33, 35, 43, 48, 49, 58, 61, 70, 74, 76, 79, 92

Z. Zhang, P. Luo, C. C. Loy, and X. Tang. Joint face representation adaptation and clustering in videos. In *European Conference on Computer Vision (ECCV)*, 2016b. xx, 2, 20, 21, 22, 23, 24, 25, 28, 29, 33, 35, 43, 47, 48, 49, 50, 58, 61, 70, 71, 74, 76, 91, 92, 93, 94, 119, 120, 121, 122

B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2017. 109

C. Zhou, C. Zhang, H. Fu, R. Wang, and X. Cao. Multi-cue augmented face clustering. In *ACM Multimedia (MM)*, 2015. 32

H. Zhou, M. Tapaswi, and S. Fidler. Now You Shake Me: Towards Automatic 4D Cinema. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 28

Y. Zhou and T. L. Berg. Temporal perception and prediction in ego-centric video. In *International Conference on Computer Vision (ICCV)*, 2015. 101