BRIDGING DISTRIBUTIONAL DISCREPANCY WITH TEMPORAL DYNAMICS FOR VIDEO UNDERSTANDING

A Dissertation
Presented to
The Academic Faculty

By

Min-Hung Chen

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology

August 2020

I dedicate this Thesis to:

My Father Ching-Shan,

My Mother Yu-Mei,

and

My Sister Yun-Ling

ACKNOWLEDGEMENTS

I would like to express my utmost gratitude to my advisor, Prof. Ghassan AlRegib, for his guidance, patience, understanding, and support through my whole Ph.D. degree. He not only guided me with constructive research directions, provided me flexibility for innovation, but also supported me in all aspects when I faced challenges and made the whole journey a productive experience. I would like to thank Prof. Zsolt Kira for broadening my view in research with a comprehensive understanding of multiple academic fields. I would also like to thank Prof. Patricio Vela, Prof. Yi-Chang Tsai, and Prof. Eva Dyer for their time, valuable comments, and for agreeing to serve as my dissertation committees.

This dissertation cannot be finished smoothly without a great working environment of the Omni Lab for Intelligent Visual Engineering and Science (OLIVES) and Center for Signal and Information Processing (CSIP). I would like to thank all my friends and colleagues within OLIVES and CSIP. Special thanks go to my dear friends and colleagues, Dr. Dogancan Temel and Dr. Tariq Alshawi for their valuable collaborations, friendships, and support over the years. I would also like to thank all CSIP faculty, staff, the academic office staff at the School of Electrical and Computer Engineering (ECE), and the Office of International Education (OIE), who provide valuable support for international students like me. Moreover, special thanks to all the collaborators during my internships at Aipoly, Sony Interactive Entertainment, and Baidu USA.

I am forever in debt to my family for their endless care, encouragement, patience, and unconditional support. My everlasting gratitude goes to my mother, Yu-Mei, for her teachings, care, and dedication. I am also eternally grateful to my father, Ching-Shan, for continuous support, care, and wise guidance. Special thanks go to my sister, Dr. Yun-Ling Chen, for her support and encouragement over the years. Furthermore, I would like to express my permanent gratitude to my wife, Yu-Chen, who is my partner, lover, and best friend, for her unconditional care, support, and dedication.

I would like to thank all my friends who supported me through this journey. You all fulfilled my life and made it colorful. Special thanks go to Dr. Chih-Yao Ma, who is my colleague, friend and, role model, for his priceless friendship, support, and collaboration throughout the whole Ph.D. adventure.

Finally, to all of my family, teachers, friends, and colleagues, I humbly thank you. Without any of you, I cannot finish this dissertation.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Video has become one of the major media in our society, bringing considerable interests in the development of video analysis techniques for various applications. **Temporal Dynamics**, which characterize how information changes along time, are the key components for videos. However, it is still not clear how temporal dynamics benefit video tasks, especially for the cross-domain case, which is close to real-world scenarios. Therefore, the objective of this thesis is to effectively exploit temporal dynamics from videos to tackle distributional discrepancy problems for video understanding.

To achieve this objective, firstly the benefits for exploiting temporal dynamics for videos are investigated, by proposing *Temporal Segment LSTM (TS-LSTM)* and *Inception-style Temporal-ConvNet (Temporal-Inception)* for general video understanding, and demonstrating that temporal dynamics can help reduce temporal variations for cross-domain video understanding. Since most previous work only evaluates the performance on small-scale datasets with little domain discrepancy, two large-scale datasets for video domain adaptation, **UCF-HMDB**$_{full}$ and **Kinetics-Gameplay**, are collected to facilitate cross-domain video research, and **Temporal Attentive Adversarial Adaptation Network (TA^{3}N)** is proposed to simultaneously attend, align, and learn temporal dynamics across domains. Finally, to utilize temporal dynamics from unlabeled videos for action segmentation, **Self-Supervised Temporal Domain Adaptation (SSTDA)** is proposed to jointly align cross-domain feature spaces embedded with local and global temporal dynamics by two self-supervised auxiliary tasks, *binary* and *sequential domain prediction*, demonstrating the usefulness of adapting to unlabeled videos across variations.

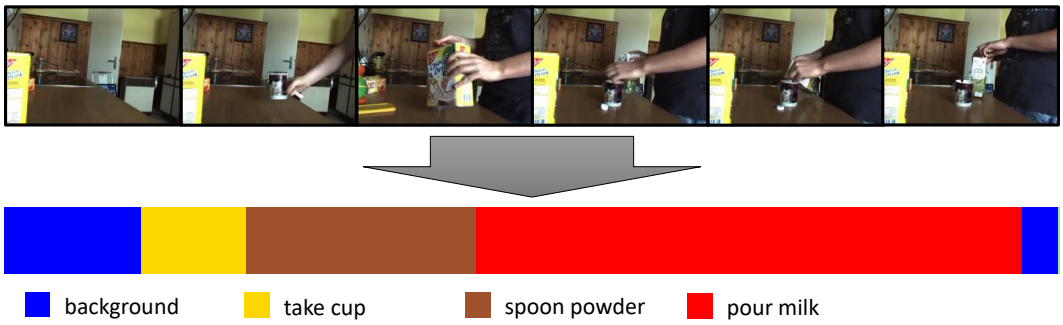
CHAPTER 1
INTRODUCTION

Video has become one of the major media in our society. The amount of videos is growing rapidly with the recent increase in the availability of video recording devices (e.g. surveillance cameras, mobile phones, etc.) and video sharing platforms (e.g. YouTube). There are already more than one billion monthly active YouTube users and six billion hours of videos are watched every month, which generates billions of views [1]. YouTube also becomes the world's second-largest search engine in the world [2]. In our daily life, we watch videos every day for many kinds of purposes, such as entertainment, communication, etc. Therefore, there are more and more related products released nowadays, such as tablets, TV, smart devices, etc. In addition to consumer products, video is also a critical component for intelligent industrial development, such as transportation, health care, education, retail, etc. All phenomena show our need for videos and this need motivates the development of video processing and analysis techniques for various purposes.

Among all the tasks, *Action Classification* is the core technology with wide applicability, including event detection, intelligent surveillance, sports video analysis, human-computer interactions, etc. This is the task of determining the type of human actions present in a given video, and the performance of video classification is positively correlated to other related tasks [3]. Different from single-image classification, the temporal correlations between image frames of a video provide additional motion information for recognition. Since most video classification datasets include human actions, *Video Classification* also refers to the same task.

In addition to the classification task, *Action Segmentation* is another fundamental task for video understanding. The goal of action segmentation is to simultaneously segment the video by time and predict each segment with a corresponding action category. In another

(a) Action Classification



(b) Action Segmentation

Figure 1.1: The overview of two main video tasks: (a) Classification and (b) Segmentation.

word, it is also a task of *Temporal Segmentation* in videos. While action classification has shown great progress given the recent success of deep neural networks [4, 5, 6], temporally locating and recognizing action segments in long untrimmed videos is still challenging.

The overview of the above two video tasks is illustrated in Figure 1.1. The output of classification tasks for a single video is a single text label, such as *Making milk* in Figure 1.1a. On the other hand, the output length of the segmentation task is the same as the input length, and each frame has its own prediction, as shown in Figure 1.1b.

Video understanding is challenging due to the highly variable nature of human actions in both spatial and temporal directions. The spatial variations imply the the reasons that cause the difference in static information, such as illumination and viewpoint. The temporal variations are closely related to movement or difference along the temporal direction, such as motion velocity and direction. Under those variations, different actions may look similar, and the same action may look differently, resulting in misunderstanding in videos. More

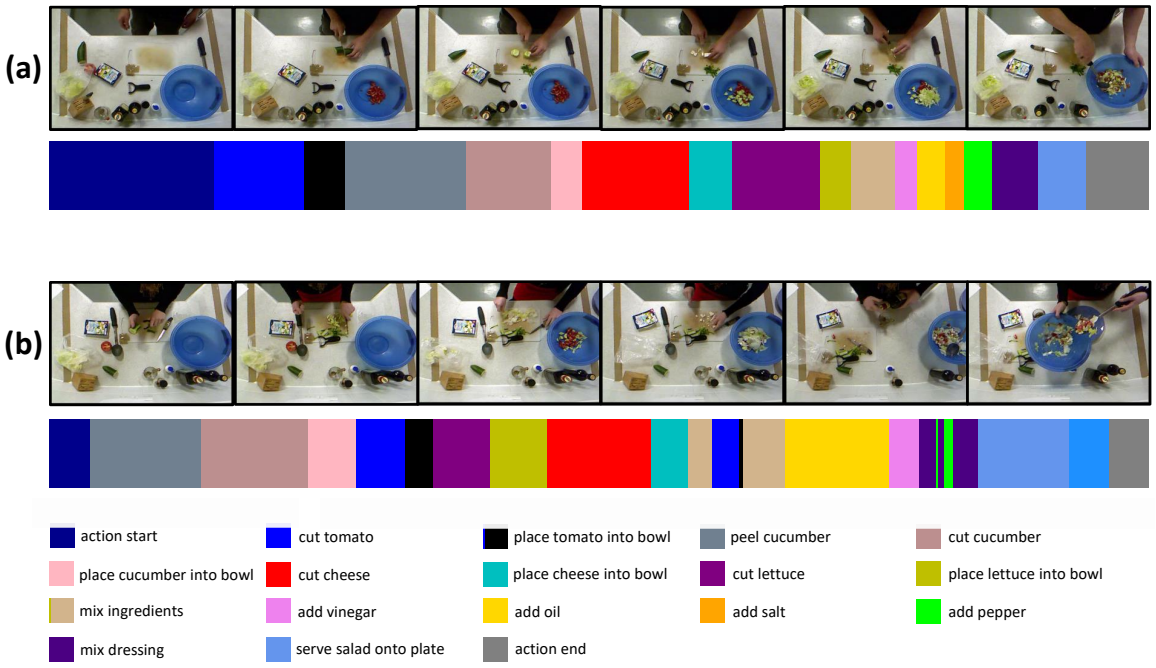


Figure 1.2: A *Making Salad* example for intra-class variation.

specifically, there are two main challenges for video understanding: *inter-class variation* and *intra-class variation* [7]. The problem of inter-class variations mainly comes from the similarities existing in different action categories. For example, *walking* and *jogging* contain similar motion patterns, which causes difficulty to distinguish classes from each other, especially when the recently proposed datasets have more than a hundred classes [8, 9, 10]. On the other hand, for the same actions, different people may behave differently in terms of styles of human motion, leading to the problem of intra-class variations. For instance, given one recipe of "making salad", different people have their own styles of movements for both spatial and temporal directions even if following the same steps of instruction. As shown in Figure 1.2, two different people are making the same salad in video (a) and (b), respectively. Despite the same activity, there is a significant difference in action labels along time. Moreover, videos in the same action can be captured from various viewpoints, showing appearance variations in different views. Finally, the tasks for video understanding are much more computationally demanding than image-based tasks since each video contains hundreds of image frames that need to be processed individually.

Encouraged by the success of using Convolutional Neural Networks (ConvNets) on still images, researchers have developed similar methods for action recognition [11, 12, 13, 14, 15, 16, 17, 18, 19, 5, 20, 21] for various video datasets [22, 23, 11, 24, 8, 9]. Most of the recent methods were inspired by two-stream ConvNets proposed by Simonyan et al. [12], which incorporate spatial and temporal information extracted from RGB and optical flow images. These two image types are fed into two separate networks, and the prediction scores from each of the streams are fused in the end. Another main research direction, starting from C3D [14], is focusing on developing ConvNet architectures to effectively extract spatio-temporal information from videos. Researchers in this direction aim to extend 2D ConvNets for videos with consideration of the trade-off between performance and computational cost.

Action segmentation approaches can be factorized into extracting low-level features using ConvNets and applying high-level temporal models. Traditionally, the temporal models are mainly recurrent models [25, 26, 27]. However, the latent state at the time step t is mainly determined by the data at t and hidden state and memory at $t-1$, which limits the capacity to learn long-range temporal dependencies. Moreover, recurrent models are known for difficult to train [28]. Therefore, encouraged by the advances in speech synthesis [29], recent approaches rely on temporal convolutions to capture long-range dependencies across frames using a hierarchy of temporal convolutional filters [30, 31, 32, 33].

One of the key components that makes videos different from images is the **Temporal Information**, which represents how information represents along the temporal direction. Temporal information, which refers to the *motion characteristics* and *temporal dynamics* for videos, is also one of the most critical factors that affect the performance of video analysis [3, 34, 7]. To exploit this critical component, several works have explored the strong use of spatiotemporal information, typically by taking frame-level features and integrating them using long short-term memory (LSTM) cells, temporal feature pooling [15, 16] and

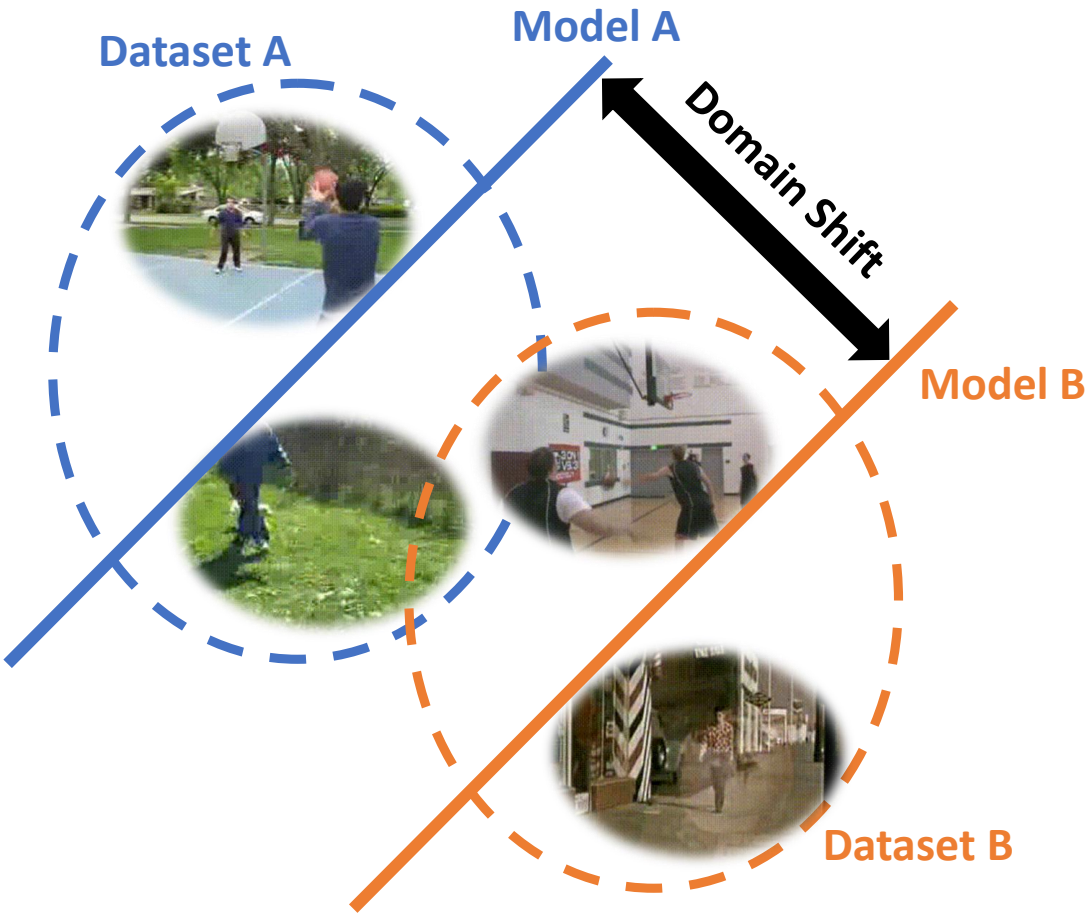


Figure 1.3: An overview of Domain Adaptation.

temporal segments [17]. However, these works typically try individual methods with little analysis of whether and how they can successfully use temporal information. Furthermore, each individual work uses different networks for the baseline approach, with varied performance depending on the training and testing procedure as well as the optical flow method used. Therefore, it is unclear how much improvement resulted from different usage of temporal information.

Another problem is that the majority of the performance gains come from the massive amount of labeled data for fully-supervised learning, including action classification and segmentation. One effective way to improve performance requires to exploit knowledge from even larger-scale labeled data [18], aiming to include more data for training to reduce the extent of inter- and intra-class variations. Since manually annotating data for the great

variety of applications and tasks is time-consuming and impractical, especially for video data, it is important to develop methods capable of leveraging large-scale labeled datasets to process unlabeled data in new domains. Unfortunately, the direct application of models learned from source data to target tasks suffers from the distributional discrepancy, which is also called *dataset shift* or *domain shift* [35, 36], causing that the models trained on source labeled dataset do not generalize well to target datasets and tasks.

The term **Domain** means *a group of data*, including raw data or feature representations. The data can be any kind of modality, such as images, videos, sound, text, etc. The learning tasks play a major role to determine whether a group of data is within the same domain, and the data share some characteristics in each defined domain. In most cases, each dataset is regarded as a single domain since all the data in a dataset are collected under the same or similar policies and protocols. There are even some datasets containing multiple domains since there exist significant distributional discrepancy problems between training and testing sets. Once a *domain* is defined, the *domain shift* simply indicates the differences across domains. Let's take Figure 1.3 as an example. In the dataset A (blue), if we want to classify two actions *Basketball* (upper) and *Walk* (lower), we can simply train a model (noted as model A) to learn an effective discriminative classifier. However, this model is not guaranteed to work on another dataset, especially when the overall distribution is very different, like dataset B (orange). And a model trained using dataset B may be far away from the first one. This is a standard illustration for the *domain shift* problem.

The main reason to cause domain shift, which is also a *distributional discrepancy* problem, is that there exist various variations between domains. Learning a discriminative model in the presence of dataset shift between training and testing distributions is known as *Domain Adaptation (DA)* [37], which has been studied extensively in recent years [38], especially after the rise of deep learning [39]. In this work, we focus on the harder unsupervised DA problem, which requires training models that can generalize to target samples without access to any target labels. While many DA approaches are able to diminish the

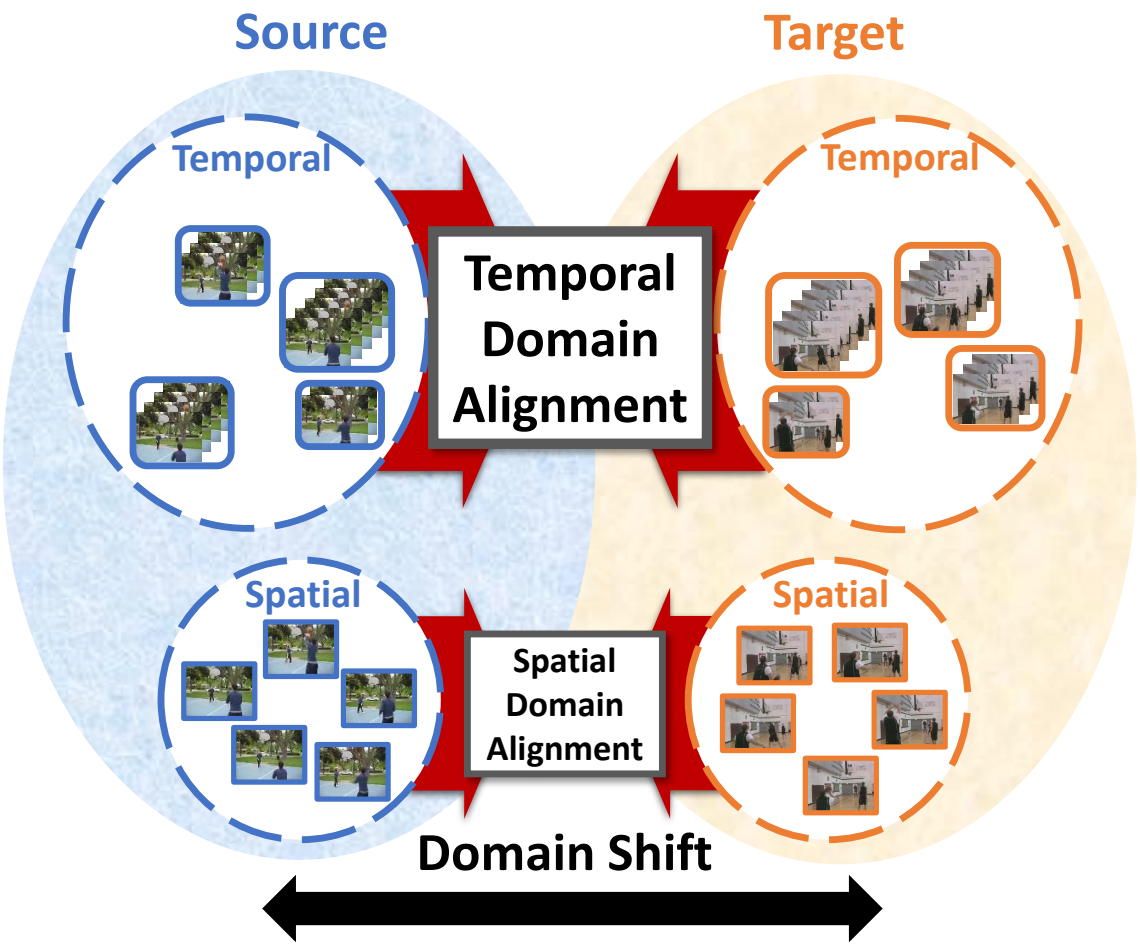


Figure 1.4: An overview of Domain Adaptation for videos.

distribution gap between source and target domains while learning discriminative deep features [40, 41, 42, 43, 44, 45, 46], most methods have been developed only for images and not videos or actions. The domain shift problem in the temporal direction in videos is not explicitly addressed using image-based DA methods, bringing the need for alignment for embedded feature spaces along the temporal direction, as shown in Figure 1.4.

To investigate the domain shift problem in videos, ideally, we should select the datasets that mainly suffer from intra-class variations. However, most datasets for classification tasks are designed for investigating inter-class variations, which means most approaches using these datasets focus on differentiating different classes. Some datasets claim that the training and testing sets are divided according to spatio-temporal variations (e.g. UCF101 [23]),

but the discrepancy problem is not significant enough so that the performance is already saturated [18]. Thus, recently several cross-domain classification datasets are proposed, focusing on the domain shift problem. However, unlike image-based DA works, which use datasets such as Office [47], VisDA [36], etc., there do not exist well-organized datasets to evaluate and benchmark the performance of DA algorithms for videos. The most common datasets are *UCF-Olympic* and *UCF-HMDB*$_{small}$ [48, 49, 50], which have only a few overlapping categories between source and target domains. This introduces limited domain discrepancy so that a deep ConvNet architecture can achieve nearly perfect performance even without any DA method. In another word, the scales of those datasets are too small to investigate domain shift problems in videos.

The goal of this work is to investigate the key factor, **Temporal Dynamics**, and how it can benefit tackling distributional discrepancy problem for video understanding under various conditions, including within- and cross-domain settings.

To achieve this goal, we first investigate the question: *given the spatial and motion features representations over time, what is the best way to exploit the temporal dynamics?* We thoroughly evaluate the design choices of the baseline two-stream ConvNet and two proposed methods, **TS-LSTM** and **Temporal-Inception**. we clarify the contribution of each decision and highlight their implication to fully exploit temporal dynamics without manipulating its inputs. We show that both proposed methods can improve significantly over baseline and achieve comparable state-of-the-art performance.

Secondly, in order to investigate the temporal domain shift problems, we propose two cross-domain datasets for action classification: 1) **UCF-HMDB**$_{full}$: we collected 12 overlapping categories between UCF101 [23] and HMDB51 [22], which is around three times larger than both the UCF-Olympic and UCF-HMDB$_{small}$ datasets. 2) **Kinetics-Gameplay**: we collected from several currently popular video games with 30 overlapping categories with Kinetics-600 [8, 9]. This dataset is much more challenging than UCF-HMDB$_{full}$ due to the significant domain shift between the distributions of virtual and real data.

With our proposed datasets, we investigate different DA integration methods for action classification and show that: 1) aligning the features that encode temporal dynamics outperforms aligning only spatial features. 2) to effectively align domains spatio-temporally, *which features* to align is more important than *what DA approaches* to use. We also propose **Temporal Attentive Adversarial Adaptation Network (TA^{3}N)** to explicitly attend to the temporal dynamics by taking into account the domain distribution discrepancy. In this way, the temporal dynamics which contribute more to the overall domain shift will be focused on, leading to more effective temporal alignment. TA^{3}N achieves state-of-the-art performance on all four investigated video DA datasets.

Finally, different from most classification datasets, most action segmentation datasets suffer from *spatio-temporal intra-class variations* of human actions across videos [7], especially when the training and validation splits are divided according to different people. To address the variation problem, we propose to diminish the distributional discrepancy caused by spatio-temporal variations by exploiting auxiliary unlabeled videos with the same types of human activities performed by different people. More specifically, to extend the framework of the main video task for exploiting auxiliary data [51, 52], we reformulate our main task as an unsupervised DA problem with the transductive setting [37, 38], which aims to reduce the discrepancy between source and target domains without access to the target labels. With the DA framework, we propose **Self-Supervised Temporal Domain Adaptation (SSTDA)**, containing two self-supervised auxiliary tasks: 1) *binary domain prediction*, which predicts a single domain for each frame-level feature, and 2) *sequential domain prediction*, which predicts the permutation of domains for an untrimmed video. Through adversarial training with both auxiliary tasks, SSTDA can jointly align cross-domain feature spaces that embed local and global temporal dynamics, to address the spatio-temporal variation problem for action segmentation, and outperforms the current state-of-the-art methods on three popular datasets: *GTEA* [53], *50Salads* [54], and *Breakfast* [55].

1.1 Contributions and Dissertation Organization

The contributions of this dissertation can be summarized as follows:

1. *Chapter 3*: We investigate and explore different ways to model the temporal dynamics of activities with feature representations from image appearance and motion, and propose two novel approaches: 1) *Temporal Segment LSTM (TS-LSTM)*: We revisit the use of LSTMs to fuse high-level spatial and temporal features to learn hidden features across time, and show that directly using LSTM performed only similar to naive temporal pooling methods, e.g. mean or max pooling, due to the limited of temporal dynamics from feature representations obtained from deep ConvNet. We further demonstrate that the integration of temporal pooling methods and LSTM yields significantly better performance. 2) *Inception-style Temporal-ConvNet (Temporal-Inception)*: We propose to use stacked temporal convolution kernels to explore temporal information at multiple scales with an Inception-style design. We show that by properly exploiting the temporal information, Temporal-Inception can also achieve comparable state-of-the-art performance even when taking feature vectors as inputs (i.e. without using feature maps).

2. *Chapter 4*: We investigate the domain shift problem across videos under the classification task. First, we organize and select videos from UCF101 and HMDB51 to form the *UCF-HMDB*$_{full}$ dataset. In addition, we also collect gameplay videos to obtain the Gameplay dataset by following the protocol of existing video datasets, and then combine with Kinetics-600 to form the *Kinetics-Gameplay* dataset. To the best of our knowledge, they are by far the largest datasets for cross-domain action classification. And then we investigate different DA integration approaches and provide a strategy to effectively align domains spatio-temporally for videos. Finally, We propose *Temporal Attentive Adversarial Adaptation Network (TA*^{3}*N)*, which simultaneously aligns domains, encodes temporal dynamics into video representations, and attends

to representations with domain distribution discrepancy, achieving achieves state-of-the-art performance on both small- and large-scale cross-domain video datasets.

3. *Chapter 5*: we reformulate the action segmentation as an unsupervised DA problem with the transductive setting, to focus on addressing the problem of intra-class variations. With the DA framework, we propose **Self-Supervised Temporal Domain Adaptation (SSTDA)**, containing two self-supervised auxiliary tasks, *binary* and *sequential domain prediction*, which predicts domains for an untrimmed video in multiple temporal scales. Through adversarial training with both auxiliary tasks, SSTDA can jointly align local and global embedded feature spaces across domains, outperforming other DA methods. Moreover, by integrating SSTDA for action segmentation, our approach outperforms the current state-of-the-art approach by large margins, and achieve comparable performance by using only *65%* of labeled training data. To the best of our knowledge, SSTDA is the first self-supervised method designed for cross-domain action segmentation.

The rest of this dissertation is organized as follows: following the motivational introduction and problem statement, we provide a literature survey discussing action classification, segmentation, and machine learning approaches beyond fully-supervised learning in Chapter 2 , aiming to provide the necessary background knowledge in this field. The novel contributions of this dissertation are introduced in Chapters 3, 4, and 5. Chapter 3 is composed of the two proposed approaches modeling the temporal dynamics: TS-LSTM and Temporal-Inception. The proposed two datasets and the novel method, TA^{3}N, are described in Chapter 4. In Chapter 5, we reformulate the action segmentation as an unsupervised DA problem, and propose a novel approach, SSTDA, which combines the advantage of domain adaptation and self-supervised learning, and can be integrated with action segmentation to tackle the spatio-temporal intra-class variation problem. Finally, the conclusion and future directions of this work are described in Chapter 6.

CHAPTER 2
LITERATURE SURVEY

In the dissertation, we aim to investigate temporal information for video understanding under various conditions, including fully-supervised and cross-domain settings. Therefore, we provide an overview and a thorough literature survey for two main research areas: *Video Understanding* and *Cross-Domain Representation Learning*.

2.1 Video Understanding

As mentioned in Chapter 1, most videos that people are interested in contain human actions, so we mainly focus on *Human Action Understanding* in this work. Among all the related tasks, *Classification* and *Segmentation* are the two core techniques and fundamental research directions for video understanding. Thus, we first provide the necessary background information and prior arts for these two tasks.

2.1.1 Action Classification

This is the task of determining the type of human actions present in a given video. Many of the action classification methods extract high-dimensional features that can be used within a classifier. These frame-level features can be hand-crafted or learned, and then combined in some form to represent videos. A common concept among these methods is that the key is how we exploit the temporal dynamics in videos [34].

Hand-crafted features: One of the main-stream hand-crafted approaches is using trajectory features (e.g. dense trajectory) [56, 57, 58] to model the motion in videos. Suppressing the noise in optical flow images has also been proven to help in action recognition [57, 58]. Fisher vectors are also adopted to generate more compact features [59, 60]. Another method is generating histogram-like features by dictionary learning [61, 62, 63]. In addi-

tion, because of the recent success of convolutional neural network, integrating learned and hand-crafted features becomes another main-stream. For example, Zhang et al. [64] modeled temporal structures with pre-trained CNN features using a linear dynamical system. Banerjee and Murino [65] used Objectness to extract local features. Conventional hand-crafted features have been undoubtedly successful in human action recognition. These approaches are able to capture local and global video descriptors and encode them in different ways to achieve state-of-the-art performance on several standard datasets[57, 58, 61, 56].

ConvNets with temporal dimension: The early work from Karpathy et al. [11] stacks consecutive video frames and extends the first convolution layer to learn the compact spatiotemporal features by designing various architectures to exploit temporal dynamics, including early fusion and slow fusion. Another proposed method, C3D [14], takes this idea one step further by replacing all of the 2D convolutional kernels with 3D kernels at the expense of GPU memory. Carreira et al. [18] further boost the accuracy by applying this idea to GoogleNet [66] instead of AlexNet [67] as in C3D [14]. To avoid high complexity when training 3D convolutional kernels, Sun et al. [68] factorize the original 3D kernels into 2D spatial and 1D temporal kernels and achieve comparable performance. Instead of using only one layer like [68], we demonstrate that multiple layers can extract temporal correlations at different time scales and provide better capabilities to distinguish different types of actions.

Recently, there are some other works proposing different CNN architectures to encode the spatiotemporal representations with consideration of the trade-off between performance and computational cost [19, 20]. Qiu et al. [19] modify 3D ResNet by factorizing 3D convolution kernel into 2D convolution (for spatial) and 1D convolution (for temporal), and integrate several residual blocks which have different arrangements of spatial and temporal convolution kernels into the final architecture P3D, showing better accuracy and less computation cost than 3D ResNet. Tran et al. [20] adopt similar factorization concepts as

P3D but use a single type of spatio-temporal residual block. With a careful choice of dimensionality, their proposed R(2+1)D outperforms P3D significantly (around 8% in video accuracy on the Sports-1M dataset). Different from the above end-to-end approaches, this dissertation mainly focuses on investigating temporal information so we separate the encoding process of the spatial and temporal information. In this way, it is more clear to see the contribution of our work for exploiting temporal information for video tasks.

ConvNets with RNNs: On the other hand, Recurrent Neural Networks (RNNs) can also be used to learn how the representations change over time for activities. Donahue et al. [13] feed spatial features extracted from each time step to a recurrent neural network with LSTM cells. In contrast to the traditional models which can only take a fixed number of temporal inputs and have limited spatiotemporal receptive fields, the proposed Long-term Recurrent Convolutional Networks (LRCN) can directly take variable-length inputs and learn long-term dependencies. Yan et al. [69] propose Hierarchical Multi-scale Attention Network (HM-AN) which incorporates the attention mechanism into hierarchical multi-scale RNN. Pan et al. [70] propose Hierarchical Recurrent Neural Encoder (HRNE) which combines the concept of temporal segments and hierarchical RNNs. They, however, validate the proposed method on video captioning tasks.

Two-stream ConvNets: Another branch of research in action recognition extracts temporal information from traditional optical flow images. This approach was pioneered by [12]. The proposed two-stream ConvNets demonstrate that the stacked optical flow images solely can achieve comparable performance despite the limited training data. Currently, the two-stream ConvNet is the most popular and effective approach for action classification since it can be further extended and combined with all the previously mentioned approaches. Ng et al. [15] take advantage of both two-stream ConvNets and LRCN, in which not only the spatial features are fed into the LSTM but also the temporal features from optical flow images. Wang et al. [71] use hand-crafted motion features, such as HOF and MBH, instead of optical flow features to boost the performance. Feichtenhofer et al. [16] investigate differ-

ent ways to fuse the spatial and temporal streams, and find that fusing the streams in the last convolutional layer using 3D convolutional kernels and 3D pooling can achieve good performance with less parameter requirement. Different from them, we show that by properly leveraging temporal information, the proposed Temporal-Inception can achieve comparable state-of-the-art results only using feature vector representations, instead of feature maps as in [16]. The work in this dissertation also shows that combining a ConvNet with vanilla LSTM results in limited performance improvement when data does not contain sufficient temporal variances.

More recent work: Similar to the above works, many other approaches exist. Wang et al. [17] propose the temporal segment network (TSN), which divides the input video into several segments and extracts two-stream features from randomly selected snippets. The authors also emphasize the importance of using pre-trained models for the temporal network to help prevent over-fitting. Wang et al. [72] incorporate semantic information into the two-stream ConvNets. Specifically, the authors incorporate ROIs (scene and person) separately into the spatial and temporal ConvNets. Each of the scene and person cues are analyzed and systematically studied. The evaluation demonstrated how each of the cues can enhance the robustness within different action types. Zhu et al. [73] propose a key volume mining deep framework to identify key volumes that are associated with discriminative actions. Wang et al. [74] propose to use a two-stream Siamese network to model the transformation of the state of the environment. Zhao et al. [75] add frame difference as the third stream and adopted VLAD [76] to encode the final features. Recently, some researchers incorporate the concepts of video object relationship [5] and video frame relationships [21] into action recognition and improve the performance. Ma et al. [5] propose to efficiently learn interactions across multiple objects for fine-grained video understanding, demonstrating that modeling object interactions significantly improves accuracy for both video classification and captioning. Zhou et al. [21] adopt the concept that human can link meaningful transformations of entities over time to propose Temporal Relation Net-

work (TRN), which learns the temporal dependencies between video frames in multiple time scales.

In this work, we aim to create a common and strong baseline two-stream ConvNet and extend the two-stream ConvNet to provide in-depth analysis of design decisions for both RNN and Temporal-ConvNet. We demonstrate that both methods can achieve comparable state-of-the-art performance but proper care must be given. For instance, LSTMs require pre-segmented data to fully exploit temporal information. The analysis in this work identifies specific limitations for each method that could form the basis of future work.

2.1.2 Action Segmentation

The goal of this task is to simultaneously segment the video by time and predict each segment with a corresponding action category. In another word, action segmentation contains both action classification and temporal segmentation. In addition to correct per-frame action classification, the correct happening time for each action segment is also critical to determine the task performance. In general, action segmentation is done by extracting low-level features using convolutional neural networks and then applying high-level temporal models. According to the types of annotation used for training, action segmentation approaches can be mainly categorized into two classes: 1) *fully-supervised* methods require that the training videos are all densely-annotated for all the frames, while 2) *weakly-supervised* methods do not need annotations for all the frames.

Fully-Supervised Approaches: Encouraged by the advances in speech synthesis [29], recent approaches rely on temporal convolutions to capture long-range dependencies across frames using a hierarchy of temporal convolutional filters [30, 31, 32, 33]. ED-TCN [30] follows an encoder-decoder architecture with a temporal convolution and pooling in the encoder, and upsampling followed by deconvolution in the decoder. However, the use of temporal pooling might result in a loss of fine-grained information that is necessary for action segmentation. TricorNet [31] replaces the convolutional decoder in the ED-TCN with

a bi-directional LSTM (Bi-LSTM). However, their network contains a temporal recurrent network, and thus inherits the limitations of recurrent models including limited attention span [25]. TDRN [32] builds on top of ED-TCN [30], using deformable convolutions instead of the normal convolution and adding a residual stream to the encoder-decoder model. MS-TCN [33] stacks multiple stages of temporal convolutional networks (TCNs) where each TCN consists of multiple temporal convolutional layers performing acausal dilated 1D convolution. With the multi-stage architecture, each stage takes an initial prediction from the previous stage and refines it. We build our approach on top of MS-TCN, focusing on developing methods to effectively exploit unlabeled videos instead of modifying the architecture.

Weakly-Supervised Approaches: Because of the difficulty of obtaining dense annotation for action segmentation, there is an increasing attention on the weakly-supervised setting where the ground truth data for training is weakly-annotated, which is much easier to obtain. HTK [77] and GRU [78] train the models in an iterative procedure starting from a linear alignment based on action transcripts. TCFPN [79] further improves the performance with a temporal convolutional feature pyramid network and a soft labeling mechanism at the boundaries. We adopt the similar concept of utilizing easy-to-obtain data focus on different goal. Instead of tackling within-domain action segmentation problems using weakly-annotated labels, we exploit unlabeled videos to adapt models to new domains with a cross-domain setting.

2.2 Cross-Domain Representation Learning

Although there is a great amount of work mentioned above showing significant progress for exploiting temporal information, those methods are evaluated on the same domain as training. The domain shift problem is not considered, which makes it challenging to apply those approaches to real-world problems. The performance of the trained models drops significantly across different domains, which is also shown in Chapter 4. Therefore, Domain

Adaptation (DA) techniques are needed to diminish this issue, achieving Cross-Domain Representation Learning.

In DA problems, there exist at least two domains: *Source* and *Target* [38]. We have full access to the source domain including data and ground truths, while we have only partial access to the target domain. In most cases, we don't have all the target ground truths. The goal of DA is to transfer the model learned with all the source data and part of target data to target domains, and aim to approach the performance of the model learned with all the target data. The DA settings are categorized in terms of the availability of annotations in the target training data. *Unsupervised* DA implies that the target data is completely unlabeled, while *Semi-supervised* DA means the target domain still has a small amount of labeled data. In this work, we focus on the harder unsupervised DA problem, which requires constructing a model that can generalize to target samples without access to any target labels during training.

Most DA approaches follow the two-branch architecture, representing the source and target branches, and aim to find the common space between the source and target domains. The models are therefore optimized with a combination of *classification loss* and a set of *domain loss* functions, which are the core techniques for different DA approaches [38].

2.2.1 Image-based Domain Adaptation

In recent years, most DA approaches are based on deep learning architectures designed for addressing the domain shift problems given the fact that deep convolutional representations significantly outperform hand-crafted features on DA problems [39]. Most visual DA approaches are evaluated and benchmarked using well-organized cross-domain image classification datasets, such as Office [47], VisDA [36], etc., and can be categorized into the following major categories:

1. *Discrepancy-based DA*: This is one of the main classes of DA methods. The metrics are designed to measure the distance between the source and target feature distri-

butions, such as variations of maximum mean discrepancy (MMD) [80, 40, 81, 82, 83, 41] and the CORAL function [84]. By diminishing the distance of distributions, discrepancy-based DA methods reduce the gap between source and target domains.

Tzeng et al. [80] propose the first method, Deep Domain Confusion (DDC), which integrates deep network with MMD, to confuse source and target domains. Long et al. improve this method by proposing the following methods: 1) Deep Adaptation Network (DAN) [40], which applies MMD to all the domain-specific layers, 2) Residual Transfer Network (RTN) [81], which adds residual connections to source classifier and introduces target entropy minimization to improve classifier adaptation, and 3) Joint Adaptation Network (JAN) [41], which aligns the joint distributions of multiple domain-specific layers by proposing the Joint MMD (JMMD) metric. In addition, Zellinger et al. [82] propose to match the higher-order central moments of probability distributions by defining a new distance function, Central Moment Discrepancy (CMD). Yan et al. [83] introduce class-specific auxiliary weights into the original MMD to tackle the class imbalance problem. Different from MMD-based approaches, Sun et al. [84] aligns the second-order statistics of the source and target distributions by the CORAL function.

2. *Adversarial-based DA*: This is another common class that adopts a similar concept as GANs [85] by integrating domain discriminators into the architectures. Through the adversarial objectives, the discriminators are optimized to classify different domains, while the feature extractors are optimized in the opposite direction. ADDA [86] uses an inverted label GAN loss to split the optimization into two parts: one for the discriminator and the other for the generator. In contrast, the gradient reversal layer (GRL) is used in some work [42, 43, 87] to invert the gradients so that the optimization can be done in one step.

   Ganin et al. propose to insert domain discriminators equipped with GRL into the

deep architecture for the main learning task. With the help of GRL during training, the main model will be optimized to maximize domain losses, and gradually confused by source and target domains. Their approach is called RevGrad [42] or DANN [43]. Zhang et al. add several domain classifiers to multiple layers and propose to select reliable pseudo-labels for target data in their final approach iCAN [87].

3. *Normalization-based DA*: It adapts Batch Normalization (BN) [88], which is also a technique that controls the distribution of feature representations, to DA problems by calculating two separate statistics, representing source and target, for normalization [44, 45, 89].

   Li et al. [44] propose AdaBN to calculate two separate sets of mean and variance for source and target domains in all Batch Normalization layers, achieving adaptation effects for DA tasks without introducing additional parameters. Carlucci et al. [89] improve AdaBN by fusing the statistics from source and target domains and automatically learning the fusing ratio to achieve more effective DA.

4. *Ensemble-based DA*: This method builds a target branch ensemble by incorporating multiple target branches in order to learn a domain-invariant feature generator under controlled variations [90, 91, 46, 92].

   French et al. [90] adopt the idea of mean teacher variant of temporal ensembling to achieve the consistency of the target branch under stochastic data augmentation. Saito et al. propose to push target data to task-specific decision boundaries by maximizing the discrepancy between two classifiers' outputs, with two variants of approaches, Adversarial Dropout Regularization (ADR) [91] and Maximum Classifier Discrepancy (MCD) [46]. Lee et al. [92] improve MCD by replacing the L1 distance metric with sliced Wasserstein discrepancy (SWD), which provides a geometrically meaningful guidance to detect target samples that are far from the support of the source and enables efficient distribution alignment.

5. *Attention-based DA*: Recently there are several methods adopting the attention mechanism to focus on different regions of images for more effective DA [93, 94].

   Wang et al. [93] propose Transferable Attention for Domain Adaptation (TADA), which contains 1) local attention for region-level domain discriminators to highlight transferable regions, and 2) global attention for image-level domain discriminator to highlight transferable images. Kurmi et al. [94] propose Certainty Attention-based Domain Adaption (CADA), which incorporates the probabilistic certainty of the discriminator from various regions while training the classifier.

   Different from the above methods, we design the attention mechanism for spatio-temporal domains, aiming to attend to the important parts of temporal dynamics for domain adaptation.

2.2.2 Video-based Domain Adaptation

Unlike image-based DA, video-based DA is still an under-explored area. Only a few works focus on cross-dataset evaluation between small-scale datasets [48, 49, 50]. Sultani et al. [48] improve the generalizability between domains by decreasing the effect of the background. More specifically, the authors propose to measure the confidence for being a foreground region in each pixel by using motion, appearance, and saliency together. Xu et al. [49] map source and target features to a common feature space using a dual many-to-one encoder architecture which is a shallow neural network. Jamal et al. [50] first propose AMLS to adapt pre-extracted C3D [14] features on a Grassmann manifold obtained by PCA, and then extend their method to a deep learning framework DAAA by utilizing GRL. However, the datasets used in these works are too small to have enough domain shift to fairly evaluate DA performance. Therefore, we propose two larger cross-domain datasets *UCF-HMDB*$_{full}$ and *Kinetics-Gameplay*, and provide benchmarks with different baseline and the proposed approaches. We also investigate DA integration approaches for videos and show that it performs better to simultaneously attends, aligns, and encodes temporal

dynamics into video features.

Moreover, some authors also proposed novel frameworks to utilize auxiliary data for other video tasks, including object detection [52] and action localization [51]. Lahiri et al. [52] adopt the concept of unsupervised adversarial image-to-image translation to transform static images to be visually confused by video frames, and use those transformed images to train video object detectors. Zhang et al. [51] propose TSRNet to transfer the knowledge from trimmed videos for improving the accuracy of untrimmed action localization using MMD. These works differ from our work by either different video tasks [52] or access to the labels of auxiliary data [51].

2.2.3 Self-Supervised Representation Learning

The goal for cross-domain representation learning is learning to adapt feature representations from one domain to others without full supervision. This goal is partially overlapped with *Self-supervised learning*, which has become popular in recent years for images and videos given the ability to learn informative feature representations without human supervision. The key is to design an auxiliary task (also noted as pretext task) that is related to the main task and the labels can be self-annotated.

Image-based Self-Supervised Learning: Image-based self-supervised learning approaches learn spatial information from images by predicting spatial characteristics within images, regarding spatial context as a source of free and plentiful supervisory signal for training [95, 96, 97].

Doersch et al. [95] predict the spatial relation between random pairs of patches from each image. The learned visual similarity from within-image context helps to enrich the feature representation for the main task. Noroozi et al. [96] predict the order of images patches after spatial shuffling, which is a Jigsaw puzzle problem. In this way, the feature mapping of object parts and their spatial arrangement are learned. Gidaris et al. [97] predict the 2D rotation angles after randomly rotating input images. By properly considering the

complexity of the auxiliary rotation prediction task, this simple approach outperforms other self-supervised methods.

Video-based Self-Supervised Learning: Most of the recent works for videos design auxiliary tasks based on spatio-temporal orders of videos [98, 99, 100, 101, 102]. These methods mainly learn spatio-temporal information by predicting the order after shuffling frames or video clips in spatio-temporal directions.

Lee et al. [98] formulate representation learning as a temporal sequence sorting task. More specifically, the authors take temporally shuffled frames as inputs and train a convolutional neural network called Order Prediction Network (OPN) to sort the shuffled sequences. Wei et al. [99] design Temporal Class-Activation Map Network (T-CAM) to learn the *arrow of time*, which indicates that a video is playing forward or backward. Kim et al. [100] combine both spatial and temporal shuffling and propose Space-Time Cubic Puzzles (ST-puzzle) to predict the spatio-temporal arrangement of 3D crops. With this task, the network can learn the spatial appearance and temporal relations simultaneously. Ahsan et al. [101] also adopt the concept of spatio-temporal puzzle but separate the process of spatial and temporal shuffling to reduce the computational and memory requirement. Xu et al. [102] extend the concept of OPN [98] using video clips instead of frames since clips are more consistent with the video dynamics. By predicting temporal orders of video clips, this approach achieves the current state-of-the-art accuracy for self-supervised video tasks.

Different from these works, our proposed auxiliary task predicts temporal permutation for cross-domain videos, aiming to address the problem of spatio-temporal variations for action segmentation. More importantly, our method generates self-annotated data across different domains to tackle domain shift problems whereas previous self-supervised learning methods only learn within-domain information.

CHAPTER 3
TEMPORAL DYNAMICS IN VIDEOS

We build on the traditional two-stream ConvNet and explore the correlations between spatial and temporal streams by using two different proposed fusion frameworks. We specifically focus on two models that can be used to process temporal data by leveraging recurrent and convolutional neural networks over temporally-constructed feature matrices: *Temporal Segment LSTM (TS-LSTM)* and *Temporal-Inception*. Both methods achieve comparable state-of-the-art performance, and we perform rigorous experimentation to elucidate which design decisions are important. Figure 3.1 schematically illustrates the proposed methods. By thoroughly exploring the space of architectural designs within each method, we can clarify the contribution of each decision and highlight its implication to fully exploit temporal information without manipulating its inputs. The work is implemented using Torch7 [103] and is publicly available[1]. For more details, please check the published paper [6].

3.1 Approaches: Temporal Segment LSTM and Temporal-ConvNet

3.1.1 Two-stream ConvNets Baseline

The two-stream ConvNets are constructed by two individual spatial-stream and temporal-stream ConvNets. The spatial-stream network takes RGB images as input, while the temporal-stream network takes stacked optical flow images as inputs. The two ConvNets are trained separately, and the probabilities estimated from the two ConvNets are fused directly for final video classification. A great deal of literature has shown that using deeper ConvNets can improve overall performance for two-stream methods. In particular, the performance from VGG-16 [104], GoogLeNet [66], and BN-Inception [88] on both spatial

[1] `https://github.com/chihyaoma/Activity-Recognition-with-CNN-and-RNN`

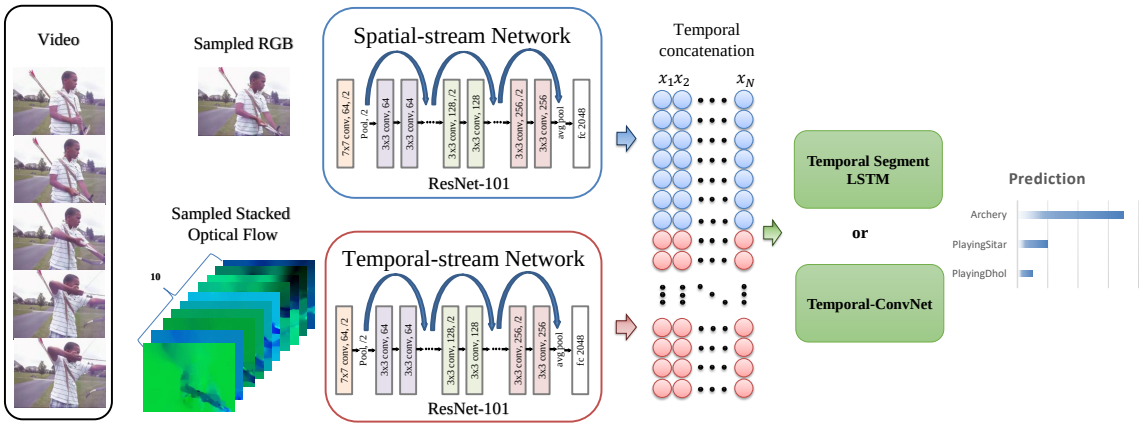


Figure 3.1: Overview of the proposed framework to learn temporal dynamics for videos.

and temporal streams are reported [17, 15]. Since ResNet [105] has demonstrated its capability in capturing still image features, we chose ResNet as the backbone ConvNet for both the spatial and temporal streams. We demonstrate that this can result in a strong baseline, as shown in Section 3.2. Feichtenhofer et al. [16] experiment with different fusion stages for the spatial and temporal ConvNets. Their results indicate that fusion can achieve the best performance using late fusions. For this reason, we aim at exploring feature fusion using the last layer from both spatial-stream and temporal-stream ConvNets. In the proposed framework, the two-stream ResNets serve as high-dimensional feature extractors. The output feature vectors at time step t from the spatial-stream and temporal-stream ConvNets can be represented as $f_t^S \in \mathbb{R}^{n_S}$ and $f_t^T \in \mathbb{R}^{n_T}$, respectively. The input feature vector $x_t \in \mathbb{R}^{n_S+n_T}$ for the proposed temporal segment LSTM and Temporal-ConvNet is the concatenation of f_t^S and f_t^T. In our case, n_S and n_T are both 2048.

Spatial stream: Using a single RGB image for the spatial stream has been shown to achieve fairly good performance. The ResNet-101 spatial-stream ConvNet is pre-trained on ImageNet and fine-tuned on RGB images extracted from UCF101 dataset with classification loss for predicting activities.

Temporal stream: Stacking 10 optical flow images for the temporal stream has been considered as a standard for two-stream ConvNets [12, 16, 15, 17, 72]. We follow the standard to show how each of our framework design and training practices can improve

the classification accuracy. In particular, using a pre-trained network and fine-tuning has been confirmed to be extremely helpful despite differences in the data distributions between RGB and optical flow. We follow the same pre-train procedure shown by Wang et al. [17]. by averaging the weights across RGB channels and replicating to the number of input channels for the temporal stream, the network can generalize better. The effectiveness of the pre-trained model on temporal-stream ConvNet is in Table 3.1 in Section 3.1.4.

3.1.2 Temporal Segment LSTM (TS-LSTM)

The variations between each of the image frames within a video may contain temporal information that could be useful in determining the human action in the whole video. One of the most straightforward ways to learn temporal dynamics from a sequence of inputs is through a Recurrent Neural Network (RNN), which maps the inputs to hidden states, and from hidden states to outputs. The objective of using RNNs is to learn how the representations change over time for actions. However, several previous works have shown limited ability of directly using a ConvNet and RNN [13, 15, 106, 24]. We conjecture that this is due to the fact that, within the same video, the sampled video frame features have similar representations. Thus, the RNNs fail to learn temporal reasoning because of the majority of the videos have similar feature representations over time, as shown in Figure 3.2. by normalizing the values of output features from the two-stream ConvNets to [0, 255], we visualize the feature representation from RGB and optical flow video frames horizontally and show how feature representation changes through time (vertical axis). As we observe from these representations, the clear vertical stripes in these three examples show that the feature representations are similar over time. We observed that the majority of the video representations in each action class in both UCF101 and HMDB51 have similar characteristics. This demonstrates the inherent lack of temporal dynamics and is the reason why vanilla RNNs show limited improvement as we discussed in Section 3.2.

Therefore, we propose to divide the sequence of inputs into several temporal segments

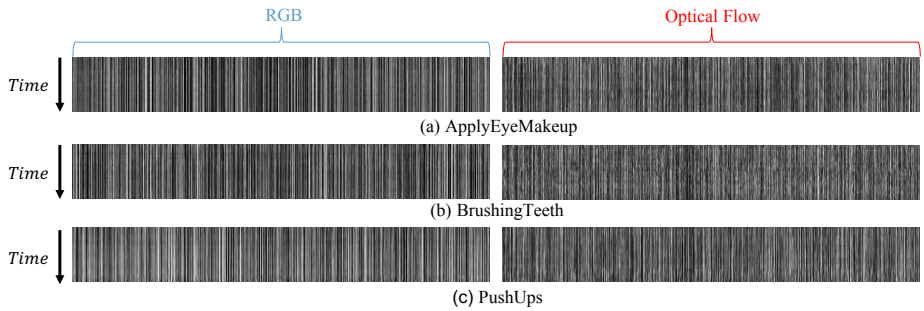


Figure 3.2: RGB and optical flow feature representations through time.

and learn their distinct feature representations. First, we concatenate the spatial and temporal features to form a high-level feature representation $x_t \in \mathbb{R}^{4096}$ for each time step. And then the concatenated features x_t are pooled within each segment through temporal pooling layers, e.g. mean- or max-pooling. The temporal pooling layer takes the feature vectors concatenated from spatial and temporal streams and extracts distinguishing feature elements. In practice, the temporal mean pooling performed similarly with max pooling. We use max pooling in our experiments. The temporally pooled features are then used as input to the recurrent LSTM module, which learns the final embedded features for the entire video. The proposed TS-LSTM module essentially serves as a mechanism that learns the non-linear feature combination and its segmental representation over time, as shown in Figure 3.3. We discuss implications of the success of TS-LSTM in Section 3.2. In practice, we evenly sampled 25 frames per video regardless the length of the video, and each temporal segment is a fixed size with $25/N$, where N is the number of temporal segments. The number of N is a hyper-parameter we chose, and we show how different N influences the overall performance in Table 3.4.

The proposed TS-LSTM share some similarity with TSN [17] in that, both of the method use temporal segments. However, the goals in utilizing temporal segments are different. While TSN use temporal segments as a way for temporal data augmentation (selecting short random snippets), we aim at using temporal segments with pooling methods to increase temporal variances, and these increased temporal variances can largely ben-

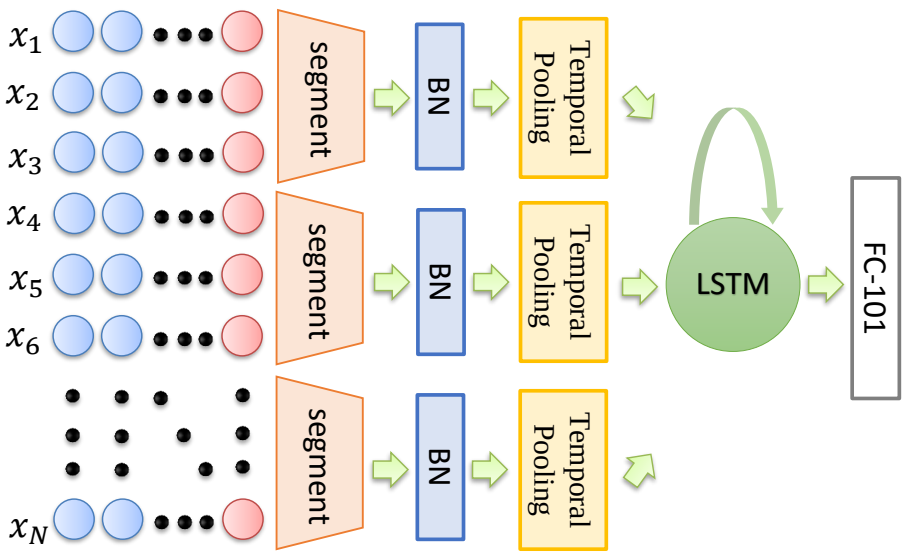


Figure 3.3: The overview of TS-LSTM.

efit standard LSTM in modeling temporal information for videos, as we demonstrate in Section 3.2.

3.1.3 Temporal-Inception

One of the main drawbacks of baseline two-stream ConvNets is that the network only learns the spatial correlation within a single frame instead of leveraging the temporal relation across different frames. To address this issue, we propose an efficient architecture with multiple convolution layers to exploit the temporal information only using frame-level feature vectors.

To address this issue, prior approaches adopted the concept of 3D kernels to exploit the pixel-wise correlation between spatial and temporal feature maps [68, 16]. However, these methods applied convolution kernels of one scale to extract the temporal features with fixed temporal receptive fields, and they did so on the full feature maps which results in more than ten times needed parameters compared to using feature vectors. In contrast to these approaches, we focus on designing a more efficient architecture with multiple convolution layers to explore the temporal information only using feature vectors (see Section 3.2.4 for more details).

In addition to using RNNs to learn the temporal dynamics, we adopt the ConvNet architecture on feature matrices $\boldsymbol{x} = \{x_1, x_2, ..., x_t, ..., x_N\} \in \mathbb{R}^{(n_S+n_T)\times N}$, where N is the number of sampled frames in one video, and t is the time step. Different from natural images, the elements in each x_t have little spatial dependency but have temporal correlation across different x_t. With the ConvNet architecture, we explore the temporal correlation of the feature elements, and then distinguish different categories of actions. To achieve this goal, we apply multiple 1D convolution kernels to specifically encode temporal information in different scales and reduce the temporal dimension. Applying convolution along the spatial direction may alter the learned spatial characteristics since the spatial information is already encoded from the two-stream ConvNets.

The overall architecture of Temporal-ConvNet is shown in Figure 3.4. We refer to a sequential layer that consists of a convolution layer, batch-normalization layer, ReLU activation layer and a pooling layer as a *Temporal-ConvNet Layer (TCL)*, which is the basic unit throughout the whole architecture. *TCL* can be expressed as follows:

$$TCL(\mathbf{x}; \mathbf{W}) = f^{pool}(\sigma(f^{BN}(\mathbf{x} * \mathbf{h_W} + b_W))) \tag{3.1}$$

where $\mathbf{h_W}$ and b_W are weights and biases of the convolutional kernel, and f^{pool}, σ, f^{BN} are pooling, ReLU and batch normalization functions, respectively.

We further effectively exploit temporal information by adapting the inception module [66] into the architecture and note it as the *multi-flow* module, which consists of different convolution kernel sizes, as shown in Figure 3.4 (each flow corresponds to one row of *TCL*s). A stack of multi-flow modules is used to hierarchically encode temporal information and reduce the temporal dimension. The rationale behind multi-flow modules is that different types of actions have different temporal characteristics, and each kernel with different size essentially searches different actions by exploiting different receptive fields to encode the temporal characteristics. With the multi-flow modules, the proposed architecture can capture actions with different scales. The overall multi-flow module can be

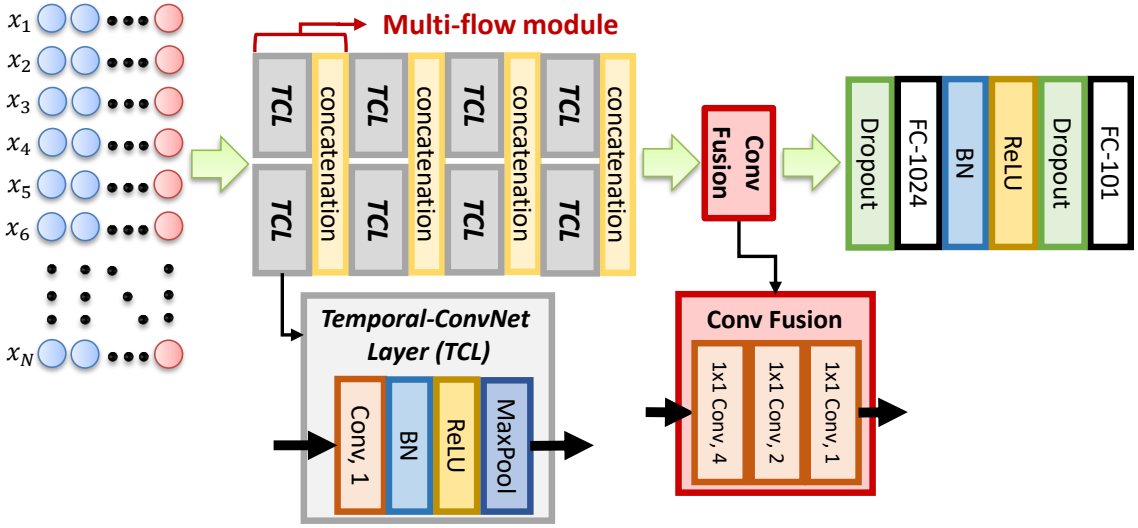


Figure 3.4: The overview of Temporal-Inception.

expressed as follows:

$$F_i(\mathbf{x}) = TCL_1(\mathbf{x}; \mathbf{W_{i_1}}) \,||\, TCL_2(\mathbf{x}; \mathbf{W_{i_2}}) \,||\, ... \tag{3.2}$$

where $||$ is the concatenation along the filter dimension, and i is the layer index. Note that in the same layer, the weights in different *TCL*s are learned from filters of different kernel sizes.

In the proposed architecture, we stack multiple multi-flow modules to reduce the temporal dimension to one (N is the number of the multi-flow modules):

$$f^{MF}(\mathbf{x}) = F_N(F_{N-1}(...(F_2(F_1(\mathbf{x}))))) \tag{3.3}$$

However, the filter dimension gradually increases because of the concatenation along filter dimension in each multi-flow module. To avoid potential over-fitting issues, we apply *Conv Fusion* to gradually reduce the filter dimension by convolving with a set of filters

$\mathbf{h}_{\mathbf{D_1}}^{\mathbf{D_2}} \in \mathbb{R}^{1\times1\times D_1\times D_2}(D_2 < D_1)$ and biases $b \in \mathbb{R}^{D_2}$:

$$f^{RD}(\mathbf{x}) = f_{d_M}^{1}(f_{d_{M-1}}^{d_M}(...(f_{d_2}^{d_3}(f_{d_1}^{d_2}(\mathbf{x})))))) \tag{3.4}$$

where $f_{d_i}^{d_j}(\mathbf{x}) = \mathbf{x} * \mathbf{h}_{\mathbf{d_i}}^{\mathbf{d_j}} + b^{d_j}$ is used to reduce the filter dimension of inputs from d_i to d_j. M is the size of the filter set.

To encode the feature vectors, we feed-forward the output of *Conv Fusion* to an 1024-dimensional fully-connected layer with the weight h_{fc} and bias b_{fc}, followed by a batch normalization layer and a ReLU layer:

$$f^{cls}(\mathbf{x}) = \sigma(f^{BN}(\mathbf{x}\mathbf{h}_{\mathbf{fc}} + b_{fc})) \tag{3.5}$$

Finally, by combining Equations (3.1) to (3.3), we can express the whole architecture as follows:

$$TemIncep(\mathbf{x}) = \rho(f^{cls}(f^{RD}(f^{MF}(\mathbf{x})))) \tag{3.6}$$

where ρ is the softmax function. We denote this architecture as *Temporal-Inception*, and show it in Figure 3.4.

The well-known state-of-the-art method from TSN [17] and the proposed *Temporal-Inception* both explore the temporal information but with different perspectives. However, TSN [17] focuses on designing novel and effective sampling approaches for temporal information exploration, while we focus on using the *Temporal-Inception* architecture to extract the temporal convolutional features given the sampled frames. We show that even with evenly-sampled frames, *Temporal-Inception* has the capability to exploit temporal dynamics between frames.

3.1.4 Implementation

Two-stream inputs: We use both RGB and optical flow images as inputs to two-stream ConvNets. For generating the optical flow images, literature have different choices for

Table 3.1: Optical flow algorithms and Temporal-stream ConvNet performance comparison on UCF101 split 1.

	Optical flow	ConvNet	Fine-tune	Accuracy
Two-stream [12]	Brox	CNN M 2048	N	81.0
Convolutional Two-stream [16]	Brox [-20,20]	VGG-M	Y	82.3
	TV-L1 [-20,20]	VGG-16	Y	86.3
Deep Two-Sream [107]	TV-L1	GoogleNet	Y	83.9
TSN [17]	TV-L1	VGG-16	Y	85.7
		BN-Inception	Y	87.2
SR-CNNs [72]	TV-L1	VGG-16	Y	85.3
Ours	TV-L1 [-20,20]	ResNet-101	Y	86.2
Ours	Brox	ResNet-101	Y	84.9

optical flow algorithms. Although most of the works used either Brox [108] or TV-L1 [109], within each of the different optical flow algorithms there are still some variations in how the optical flow images are thresholded and normalized. We summarize the prediction accuracy of different optical flow methods from recent works in Table 3.1. Note that we thresholded the absolute value of motion magnitude to 20 and rescale the horizontal and vertical components of the optical flow to the range [0, 255] for TV-L1. From Table 3.1, we can conclude that both Brox and TV-L1 can achieve state-of-the-art performance, but from our experiments TV-L1 is slightly better than Brox. Thus, unless specified we use TV-L1 as input for the temporal-stream ConvNet.

Hyper-parameter optimization: The learning rate of the spatial-stream ConvNet is initially set to 5×10^{-6}, and divided by 10 when the accuracy is saturated. The weight decay is set to be 1×10^{-4}, and momentum is 0.9. On the other hand, the learning rate of the temporal-stream ConvNet is initially set to 5×10^{-3}, and divided by 10 when the accuracy is saturated. The weight decay and momentum are the same as the spatial-stream ConvNet. The batch sizes for both ConvNets are 64. Both TS-LSTM and Temporal-ConvNets are trained with ADAM optimizer. The learning rate is set to 5×10^{-5} for training TS-LSTM. For Temporal-ConvNets, we use 1×10^{-4} for learning rate and 1×10^{-1} for weight decay.

Data augmentation: Data augmentation has been very helpful especially when the training data are limited. During training, a sub-image with size 256 $\times$ 256 is first randomly cropped using a smaller image region (between 0.08 to 1 of the original image area). Second, the cropped image was randomly scaled between 3/4 and 4/3 of its size. Finally, the cropped and scaled image will be scaled again to 224 x 224. The same data augmentation is applied when training both spatial- and temporal-stream ConvNets. Note that we use additional color jittering for the spatial-stream ConvNet, but not temporal-stream ConvNet. Wang et al. [17] argued that a corner cropping strategy, which only crops 4 corners and center of the image, can prevent over-fitting. However, empirically, we found out that the corner cropping strategy can make the ConvNet converge faster and leads to over-fitting.

Testing: We followed the work from Simonyan and Zisserman [12] to use 25 frames for testing. Each of the 25 frames is sampled equally across each of the videos. During testing, many works averaged predictions from the RNN on all 25 frames. We did not average the prediction because it will be biased towards the average representation of the video and neglect the temporal dynamics learned by LSTM cells. However, without temporal segments, since LSTM cells fail to learn the temporal dynamics, it will benefit from averaging the predictions. Yet, the final prediction accuracy is significantly worse than TS-LSTM without averaging prediction.

3.2 Experiments and Discussions

We validate the proposed models on two of the most widely used action recognition benchmarks: UCF101 [23] and HMDB51 [22]. We use the first split of UCF101 to validate the proposed models, and then keep the same parameters for the other two splits. For fair comparison, we uniformly sample 25 frames for each video.

3.2.1 Datasets

UCF101 contains 13320 videos from 101 action categories. The videos have spatial resolution of 320 $\times$ 240 pixels and the average length of 7 seconds, which implies a total of 2.4M frames in the whole dataset. Videos in each action category are grouped into 25 groups. Each group consists of 4 to 7 videos of a particular action. The videos from the same group are similar in terms of some characteristics, such as environments, camera movement, etc. Therefore, the intra-class variations across the videos in the same group are smaller than the variations between different groups. Thus, the dataset is separated into three different train/test splits. In each split, 18 groups are used for training, and the rest 7 groups are used for testing.

HMDB51 contains 6849 videos, which are divided into 51 action categories, each containing a minimum of 101 videos. The videos are extracted from a variety of sources ranging from digitized movies to YouTube. The model pre-trained on UCF101 is fine-tuned for the HMDB51 dataset.

3.2.2 Experimental Results: Two-stream ConvNets Baseline

It is important to have a fair comparison with other methods which share the same baseline or similar baseline performance. Therefore, we explicitly show, in Table 3.2, our baseline performance for the spatial-stream, temporal-streams, and two-stream on three different splits in the UCF101 and HMDB51 datasets. Our performance on the two-stream model is obtained by taking the mean values of the prediction probabilities from both spatial and temporal-stream ConvNets. For comparison of our baseline method with others, please refer to Table 3.2.

Using this baseline, we will then demonstrate how each of the proposed method leverages the baseline two-stream ConvNet and show significant improvement by modeling temporal dynamics. We also study the effectiveness of the length of the videos on the baseline performance and the two proposed methods. For further details, please see Section 3.2.6.

Table 3.2: Performance from spatial and temporal-stream ConvNets, and two-stream ConvNet on three different splits of the UCF101 and HMDB51 datasets.

	Spilit 1	Split 2	Split 3	Mean
Spatial-stream				
UCF101	86.1	83.6	85.3	85.0
HMDB51	51.9	49.7	49.7	50.4
Temporal-stream				
UCF101	86.2	87.0	88.4	87.2
HMDB51	60.3	59.0	60.0	59.7
Two-stream				
UCF101	92.6	92.2	92.9	92.6
HMDB51	66.4	64.8	62.6	64.6

Table 3.3: Two-stream ConvNet comparison on the UCF101 dataset (split 1).

	Spatial	Temporal	Two-stream
GoogLeNet [107]	77.1	83.9	89.0
VGG-16 [107]	79.8	85.7	90.9
BN-Inception [17]	84.5	87.2	92.0
ResNet-101	86.1	86.2	**92.6**

Baseline Comparison on UCF101: A great deal of literature has shown that using deeper ConvNets can improve overall performance for two-stream methods. However, to our best knowledge, there is no comparison of how different ConvNets perform as the baseline. Thus, we summarize their baseline performance in Table 3.3. In particular, the performance of VGG-16 [104], GoogLeNet [66], and BN-Inception [88] on both spatial and temporal streams are reported [17, 15]. In this particular experiment, we demonstrate a slightly better baseline performance using ResNet-101 [105]. Using this baseline, we will then demonstrate how each of the proposed methods leverages the baseline two-stream ConvNet and show significant improvement by modeling temporal dynamics.

3.2.3 Experimental Results: TS-LSTM

Through rigorous experiments, we conclude that: (i) using temporal segments performs better than no segments. (ii) LSTMs effectively learn the spatial and temporal feature integration with temporal segments while reducing the feature dimension. (iii) deep LSTM

layers do not necessarily help and often lead to over-fitting. These experiments lead us to conclude that despite their theoretical ability to extract temporal patterns over multiple scales and lengths, in this case, their inputs must be pre-segmented, demonstrating a limitation of the current usage of LSTMs. The summarization of the experiments on TS-LSTM is shown in Table 3.4.

Temporal segments and pooling: Temporal segments have been shown to be useful with temporal augmentation in end-to-end frameworks [17]. In the experiments, we demonstrate that even with pre-extracted feature vectors from equally sampled frames in a video, temporal segments can help in improving the classification accuracy. The difference between using three or five segments is statistically insignificant and may highly depend on the type of action performed in the video.

Spatial and temporal feature integration: Since the features from each segment are concatenated together, the model suffers from over-fitting when the number of segments increases, as can be seen from Table 3.4 with five segments. By adding an FC layer before the temporal max pooling layer, we can effectively control the dimensionality of the embedded features, exploit the spatial and temporal correlation, and mitigate over-fitting.

Vanilla LSTM: LSTM cells have the ability to model temporal dynamics, but only shown limited improvement from previous works. The experimental results show that there is only a 0.2% improvement over two-stream ConvNet, which is consistent with [15] (88.0% to 88.6%) and [13] (69.0% to 71.1% on RGB, 72.2% to 77.0% on flow), and [24] (63.3% to 64.5%). The performance gained from LSTM cells decreases when the baseline two-stream ConvNet is stronger in robustly representing video frames. Thus, we observe only 0.2% improvement when using a baseline two-stream ConvNet achieving 92.6% accuracy, as opposed to 2%-5% improvement from a baseline of 70% accuracy [13]. Note that using vanilla LSTM performs only similar to naive temporal max-pooling.

To further understand why using vanilla LSTM only has limited improvement, we analyze the variance of feature representations across time from both RGB and optical flow

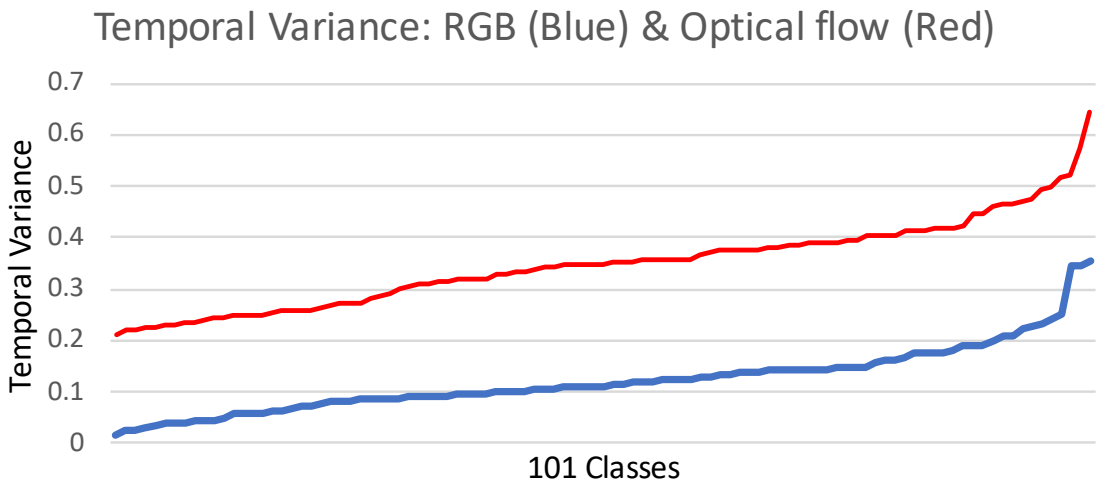


Figure 3.5: Temporal variance of feature representations across time in UCF101.

modalities as shown in Figure 3.5. As we can see that, the majority of the video representations have low variance across time. This is significantly different from how we typically used RNNs to encode or decode signals for speech and natural language processing.

TS-LSTM: By combining the temporal segments and LSTM cells, we can leverage the temporal dynamics across each temporal segment, and significantly boost the prediction accuracy. The experimental results also indicate that using deeper LSTM layers is prone to overfitting. This is probably because the features generated from spatial and temporal-stream ConvNets were fine-tuned to identify video classes at the frame level. The dynamics of feature representations over time is not complicated enough for LSTM. Thus, increasing the number of stacked LSTM layers results in overfitting. Our experimental results indicate that using deeper LSTM layers is prone to overfitting on the UCF101 dataset. This is probably because our features generated from spatial and temporal ConvNets were fine-tuned to identify video classes at the frame level. The dynamics of feature representations over time is not as complicated as other sequential data, e.g. speech and text. Thus, increasing the number of stacked LSTM layers tends to overfit the data in UCF101. Our experiments on HMDB51 also confirm this hypothesis. The prediction accuracy on the HMDB51 dataset using two-layer LSTM increased from 68.7% to 69.0% over single LSTM layer, since the

Table 3.4: Complete Performance of different architecture choices for TS-LSTM.

TS	BN	Temporal Pooling	BN	FC	Acc
LSTM					
1	BN	512	BN	101	92.8
1	BN	(512,512)	BN	101	92.5
1	BN	(512,512,512)	BN	101	92.1
Temporal segment & Batch normalization					
1	BN	Max	BN	101	92.8
3	BN	Max	BN	101	93.4
5	BN	Max	BN	101	93.2
Temporal Segment LSTM					
3	BN	Max + 512	BN	101	**94.3**
3	BN	Max + (512,512)	BN	101	**94.2**
3	BN	Max + (512,512,512)	BN	101	93.9

baseline model has not yet learned to robustly identify video classes at the frame level.

The above findings suggest that carefully re-thinking and understanding how LSTMs model temporal information is necessary.

Note that Pan et al. [70] proposed Hierarchical Recurrent Neural Encoder (HRNE) which combines the concept of temporal segments and hierarchical RNNs. Both of their method and the proposed TS-LSTM share the merit of having hierarchical structure for the RNN. However, using LSTM as first layer is still unable to extract distinct features from the repetitive sequence representation. To verify our hypothesis, we further adapted HRNE on UCF101, yet the results are only comparable when introducing FC layer within temporal segments.

Overall, among the 15 classes in UCF101 with highest temporal variance for both RGB and optical flow feature representations (see Figure 3.6), we demonstrate significant top-1 accuracy improvement particularly in *HighJump* (62% to 97%) and *PizzaTossing* (67% to 91%). This is mainly because our TS-LSTM is able to *summarize* the video representations for each temporal segment, and the following LSTM is thus able to reason through time.

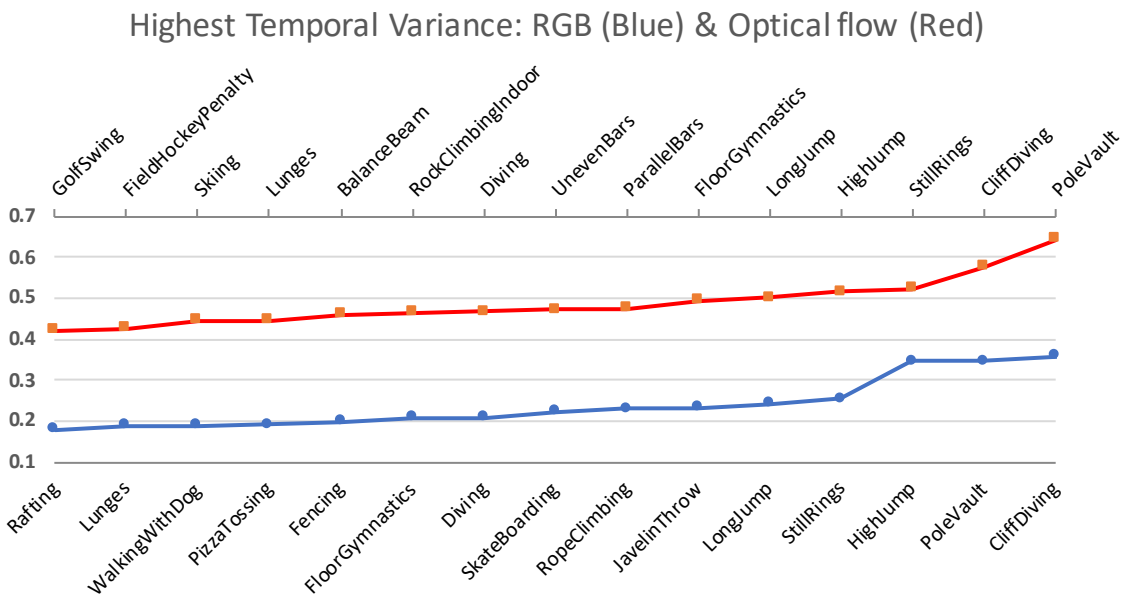
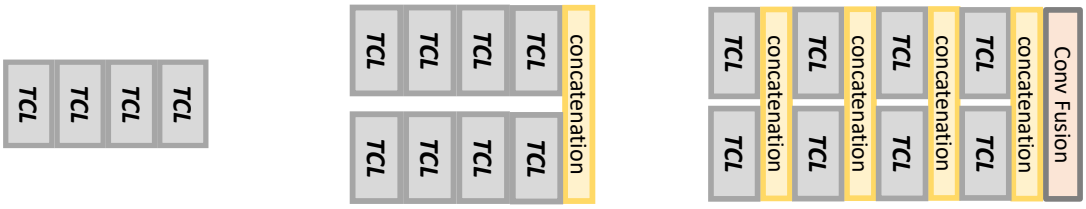


Figure 3.6: 15 classes in UCF101 with highest temporal variance. The numbers are calculated from normalized features to emphasize the difference between classes and sorted in an ascending order.

3.2.4 Experimental Results: Temporal-Inception

Here we discuss different factors for designing the architecture of the Temporal-ConvNet. We conclude that: (i) applying multiple *TCL*s performs better than using single or double *TCL*s. (ii) concatenating the outputs of each multi-flow module is the better way to fuse different flows. (iii) with proper fusion methods, the multi-flow architecture has better capability to explore the temporal information than the single-flow architecture. The summarization of the experiments on Temporal-ConvNet is shown in Table 3.6. In the column "Architecture", "T" is denoted as one *TCL*. {} denotes as the stacked architecture, while () denotes as the wide (parallel) architecture. Based on these experiments, we can conclude that convolution across time can effectively extract temporal patterns, but as with other applications the specific architecture is crucial and lessons learned from other architectures and tasks can inform successful designs.

Temporal-ConvNet layer: One of the most important components in the Temporal-ConvNet is the *TCL*, and the number of *TCL*s plays an important role, because the *TCL*s are



(a) Temporal-VGG (b) Multi-flow Temporal-VGG (c) Temporal-Inception

Figure 3.7: Comparison of three different architectures of Temporal-ConvNet.

used to gradually reduce the dimension in the temporal direction, i.e., map the feature matrices to a feature vector. Applying only single or double *TCL*s will still leave the temporal direction to have a high dimensionality, and thus result in larger numbers of parameters and cause over-fitting. We use multiple *TCL*s to avoid this issue, so the performance boosts as expected.

Multi-flow architecture: There are three main questions in optimizing the multi-flow architecture: *1) How to combine multiple flows? 2) How many flows should we have? 3) What are the best kernel sizes?*

For the first question, we propose two approaches. One is the proposed *Temporal-Inception*, and the other one is the multi-flow version of *Temporal-VGG*. The difference between these two approaches is where we place the concatenation layers, as shown in Figure 3.7. We show that *Temporal-Inception* can effectively exploit and combine temporal information obtained through various temporal receptive fields, and properly increases the accuracy. The illustration of *Temporal-VGG*, *Multi-flow Temporal-VGG*, and *Temporal-Inception* are shown in Figure 3.7.

Regarding the second question, increasing the flow number provides a better capability to describe actions with different temporal scales, but it also greatly increases the chance to overfit the data. Finally, the sizes of the temporal convolutional kernels are critical since they directly affect how the network learns the temporal correlation. The different kernel sizes essentially correspond to actions with different temporal duration and period. In *Temporal-Inception*, kernel sizes of 5 and 7 achieve the best prediction accuracy.

Table 3.5: Dimension reduction methods for Temporal-ConvNet. The performance is shown on UCF101 split1. Conv1, n: convolution with the kernel size 1×1 and the output filter dimension is n.

Reduce the dimension for each multi-flow module	
Average pooling	92.8
Max pooling	92.9
Conv fusion (Conv1, 1)	92.8
Reduce the dimension after all the multi-flow modules	
Average pooling	93.0
Max pooling	93.1
Conv fusion (Conv1, 4 - Conv1, 2 - Conv1, 1)	**94.2**

As shown in Table 3.6. *Temporal-Inception* has better performance than the multi-flow version of *Temporal-VGG*, since *Temporal-Inception* effectively exploits and combines temporal information obtained through various temporal receptive fields.

Filter dimension reduction: Although *TCL*s can reduce the temporal dimension, the filter dimension will increase because of the concatenation of multi-flow modules. There are two different approaches to reduce the filter dimension: (i) reduce the dimension for each multi-flow module (ii) reduce the dimension after all the multi-flow modules. Average pooling, max pooling and Conv fusion (convolve with a set of filters to reduce the filter dimension.) are used as the dimension-reduction methods. Table 3.5 shows the experiment results. The accuracy drops dramatically if applying the above methods for each module because we partially lose information for each dimension-reduction process. We also found that using multiple convolution layers to gradually reduce the dimension is better than directly applying average or max pooling.

Feature map vs. feature vector: To address the issue of ignoring temporal correlation, prior approaches adopted the concept of 3D kernels to exploit the pixel-wise correlation between spatial and temporal feature maps [68, 16]. However, these methods mapped high-dimensional feature maps to fully-connected layers, which cause large parameter numbers. [68] mapped $5 \times 5 \times T$ (frame number) $\times 2 \times 32$ feature maps to 4096-dimensional feature vectors, and then forwarded to another two 2048-dim fully-connected layers. [16] applied

Table 3.6: Complete Performance of different architecture choices for Temporal-ConvNet.

Architecture	BN	Dropout	FC	Accuracy
single *TCL*				
T	BN	Dropout	1024	93.6
(T,T)	BN	Dropout	1024	92.6
double *TCLs*				
{T,T}	BN	Dropout	1024	93.1
{(T,T),(T,T)}	BN	Dropout	1024	92.7
Temporal-VGG				
{T,T,T,T}	BN	Dropout	1024	94.0
Multi-flow Temporal-VGG				
({T,T,T,T},{T,T,T,T})	BN	Dropout	1024	93.4
Temporal-Inception				
{(T,T),(T,T),(T,T),(T,T)}	BN	Dropout	1024	**94.2**

the $3 \times 3 \times 3$ max-pooling layer to $7 \times 7 \times T \times 512$ feature maps, and then mapped to 512-dimensional feature vectors. There are also another two fully-connected layers before the final output. On the contrary, we only map 4096-dimensional feature vectors to 1024 dimension, which results in much less parameters needed. While [16] has totally 97M parameters, our architecture only needs 4M parameters.

3.2.5 Final Performance

We summarize the results from the proposed TS-LSTM and Temporal-Inception on both UCF101 and HMDB51 in Table 3.7. Our proposed methods achieved comparable state-of-the-art performance by modeling the temporal dynamics extracted at only 25 video frame feature representations obtained from each video. We have proven that even training with consistent high-level abstract feature representations can still achieve comparable state-of-the-art performance. Specifically, our TS-LSTM explore temporal segments and demonstrate that, a simple temporal pooling layer and LSTM cells trained on a fixed and sampled video frame representations can perform comparably with other state of the arts, even when they were trained in an end-to-end fashion, e.g. TSN [17], ST-Multiplier [110], ST-Pyramid Network [111], and TLE [112]. Note that we consider our method complementary

to TSN [17] and might be further improved if we adapt temporal random sampling.

3.2.6 Ablation Study and Analysis

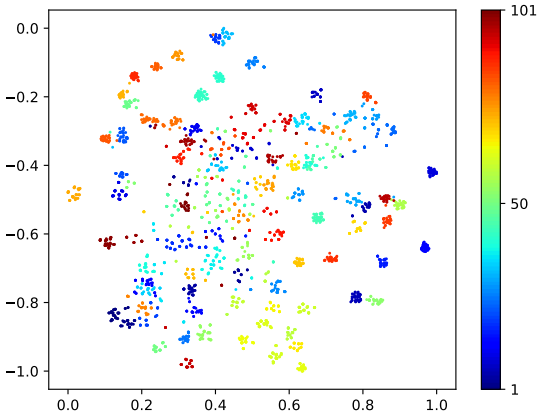
Modeling Temporal Dynamics: To further validate that our methods can model the temporal dynamics, we train both TS-LSTM and Temporal-Inception using only a maximum of the first 10 seconds of videos, e.g. 250 frames per video. Note that the number of frames in the UCF101 dataset ranges from 30 to 1700. About 21% of videos are longer than 10 seconds.

Our experiments in Table 3.8 show that by only using the first 10 seconds of the video, the baseline two-stream ConvNets achieve 92.9% accuracy which is slightly better than when using the full length of the videos. On the other hand, when the proposed methods see more frames of each video, both of the methods achieved better prediction accuracy (TS-LSTM: 93.7 to 94.1%; Temporal-Inception: 93.2 to 93.9%). This verifies that our proposed methods can successfully leverage the temporal information.

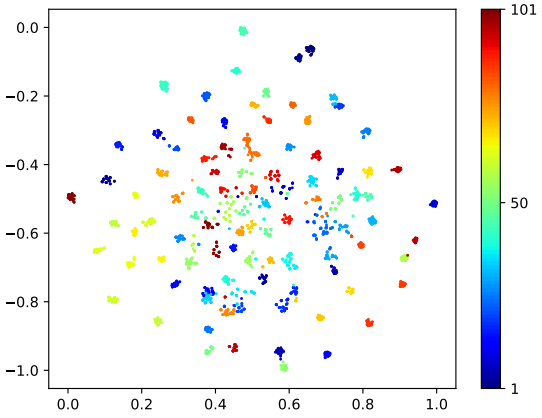
t-SNE Visualization: In addition to the prediction accuracy, we can also visualize the distance between each of the video feature representation before the last fully-connected layer. Ideally, the feature representations for videos belong to the same action class should be very closed to each other compared to feature representations from other action classes.

We thus visualize the feature vectors from baseline two-stream ConvNet, TS-LSTM, and Temporal-Inception methods using t-SNE visualization method. From Figure 3.8, we can see that both of the proposed methods are able to group the test samples into more distinct clusters. Thus, after the last fully-connected layer, both the proposed methods can achieve better classification results.

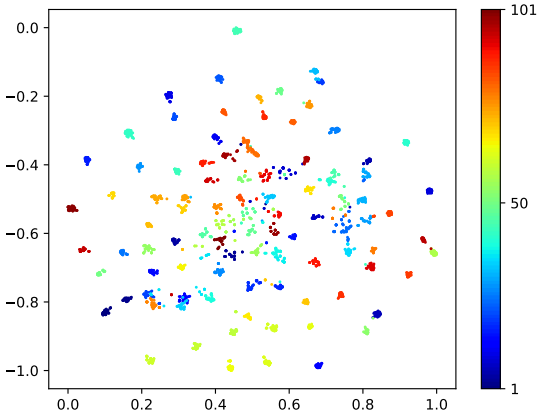
To demonstrate how the proposed method improved upon the baseline method on specific action classes, we identify the two action classes in UCF101, which have significant difference in prediction accuracy. We qualitatively show how they were misclassified by the baseline approach but correctly predicted by the two proposed methods. Specifically,



(a) Two-stream ConvNet



(b) TS-LSTM



(c) Temporal-Inception

Figure 3.8: t-SNE visualization of the last feature vector representation.

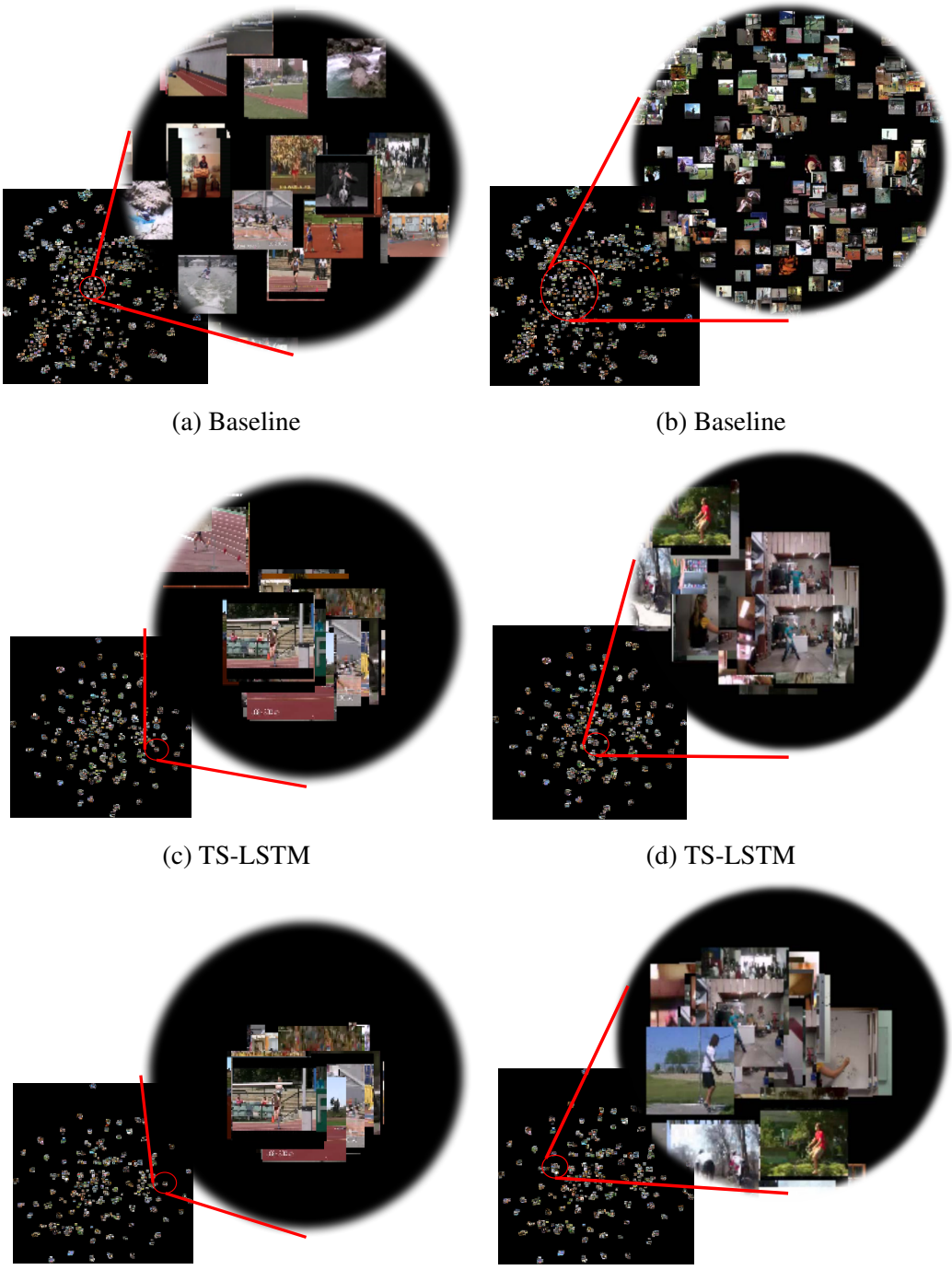
the two action classes are *HighJump* and *PizzaTossing*, both with 30% improved prediction accuracy by the proposed methods.

By superimpose the snap image from each of the video on the data points optimized from t-SNE and zoom-in these figures, as shown in Figure 3.9, it is clear that the proposed TS-LSTM and Temporal-Inception further pushed the data points belong to the same class toward each other in the high-dimensional space. For example, both *HighJump* and *PizzaTossing* videos were originally scattered around the center of the figures, but were able to grouped together by the two proposed methods. Specifically, the accuracy of individual methods are: 62.2% (baseline), 97.3% (TS-LSTM), and 94.6% (Temporal-Inception) for *HighJump*; 66.7% (baseline), 90.9% (TS-LSTM), and 97.0% (Temporal-Inception) for *PizzaTossing*.

Qualitative Analysis on UCF101: To further demonstrate how the proposed method improves upon the baseline method, we use the action class *HighJump* to qualitatively show how they were misclassified by the baseline approach but correctly predicted by the two proposed methods.

For *HighJump*, all the videos in this category can be divided into two parts: running and jumping. By applying the baseline approach, some videos are misclassified as other categories including running and jumping. In Figure 3.10, four "HighJump" example videos are shown in the upper row. The baseline approach misclassifies Figures 3.10a and 3.10b as *JavelinThrow* and *LongJump*, respectively. Figures 3.10c and 3.10d are misclassified as *FloorGymnastics* and *PoleVault*, respectively. Examples from these incorrectly predicted actions are shown in the lower row. However, those examples are correctly classified by TS-LSTM and Temporal-Inception. This shows both proposed approaches can effectively extract the temporal information.

In Figure 3.11, we show three different variations of this category. Figure 3.11(a) is misclassified as *Punch* since both videos include two people and their arm motion are also similar. Figure 3.11(b) is misclassified as *Nunchucks* because the people in both videos are

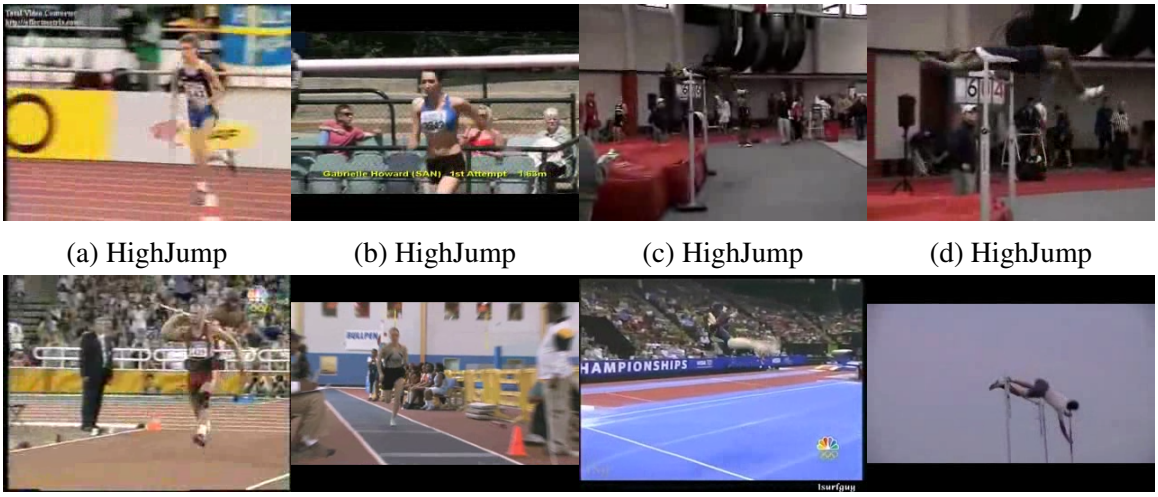


(a) Baseline (b) Baseline

(c) TS-LSTM (d) TS-LSTM

(e) Temporal-Inception (f) Temporal-Inception

Figure 3.9: The t-SNE visualization of baseline two-stream ConvNet, TS-LSTM, and Temporal-Inception on UCF101 split 1 (*HighJump*: (a)(c)(e), *PizzaTossing*: (b)(d)(f),).



(a) HighJump (b) HighJump (c) HighJump (d) HighJump

(e) JavelinThrow (f) LongJump (g) FloorGymnastics (h) PoleVault

Figure 3.10: The category (*HighJump*) that is misclassified by the baseline approach but correctly classified by TS-LSTM and Temporal-Inception.

making an object spinning. Finally, Figure 3.11(c) is misclassified as *SalsaSpin* since in both videos, there are two objects spinning quickly. Those three examples are also correctly classified by both of our approaches because we take the spatial and temporal correlation into account to distinguish different categories.

(a) PizzaTossing (type 1)(b) PizzaTossing (type 2)(c) PizzaTossing (type 3)

(d) Punch (e) Nunchucks (f) SalsaSpin

Figure 3.11: Another category (PizzaTossing) that is misclassified by the baseline approach but correctly classified by TS-LSTM and Temporal-Inception.

Learnable Pooling Methods: In the proposed TS-LSTM, we demonstrate that how a

simple max or mean pooling method combined with LSTM can further improve the video classification accuracy. On the other hand, there have been recent work shown a learnable pooling layer can be used for various image or video understanding tasks [121, 122, 123, 5, 124]. For instance, Ma et. al [5] compared the use of self-attention mechanism with traditional LSTM and shown that learnable self-attention mechanism performs better than directly using LSTM on video classification task. On the other hand, Long et al. [124] thoroughly investigated various attention mechanisms and their integration with a multi-modalities model. Certainly, the integration of attention methods with RNN is an open-research problem as recently be investigated in the Natural Language Processing field [125].

End-to-end Architecture: In addition to using pre-extracted feature vectors as the inputs, we also demonstrate our proposed TS-LSTM and Temporal-Inception in an end-to-end fashion. We use ResNet-50 as the backbone architecture and test the performance using the first split of RGB stream on the HMDB51 dataset. We show that both our proposed approaches also outperform the baseline approach and Temporal Segment Networks [17] in an end-to-end fashion, as shown in Table 3.9,

Table 3.7: State-of-the-art action recognition comparison on the UCF101 [23] and HMDB51 [22] datasets.

Methods	UCF101	HMDB51
LCRN [13]	82.9	-
LSTM composite [113]	84.3	-
$F_{st}CN$ [68]	88.1	59.1
Pairwise trajectory [60]	88.1	63.3
C3D [114]	86.7	-
Two-stream [12]	88.0	59.4
Two-stream dictionary [62]	-	59.5
Salient trajectory [115]	88.2	60.0
MIDDL [59]	-	60.3
Pyramid pooling [71]	-	60.8
LSTM [15]	88.6	-
TDD [116]	91.5	65.9
Trajectory pooling [75]	92.1	65.6
Tube ConvNet [117]	92.3	-
Saliency Context [118]	92.4	-
Transformation [74]	92.4	63.4
Convolutional Two-stream [16]	92.5	65.4
Multi-stream [119]	92.6	-
SR-CNN [72]	92.6	-
Key volume [73]	93.1	67.2
Action-vector [63]	93.0	-
ST-ResNet [120]	93.4	-
TSN (2 modalities) [17]	94.0	68.5
ST-Multiplier [110]	94.2	68.9
ST-Pyramid Network [111]	94.6	68.9
TLE [112]	**95.6**	**71.1**
TS-LSTM	94.1	69.0
Temporal-Inception	93.9	67.5

Table 3.8: Action recognition using a maximum of 10 seconds (250 frames) of each video in UCF101.

UCF101 [23]	10 seconds	Full video
Two-stream ConvNet	92.9	92.6
TS-LSTM	93.7	94.1
Temporal-ConvNet	93.2	93.9

Table 3.9: End-to-end network comparison on the HMDB51 dataset (RGB split 1).

Baseline	TSN [17] (3 segments)	TS-LSTM (3 segments)	Temporal-Inception
53.9	54.4	57.5	59.1

CHAPTER 4

CROSS-DOMAIN VIDEO CLASSIFICATION

The objective of the proposed research is to address the domain shift problems in videos, including spatial and temporal domain adaptation, as illustrated in Figure 1.4. To achieve this goal, we present two large datasets for video domain adaptation, *UCF-HMDB*$_{full}$ and *Kinetics-Gameplay*, including both real and virtual domains. We use these datasets to investigate the spatial and temporal domain shift problems across videos, and show that aligning the features that encode temporal dynamics can improve the overall performance compared to only aligning spatial features. We also propose **Temporal Attentive Adversarial Adaptation Network (TA^{3}N)** to simultaneously attend, align, and learn temporal dynamics across domains, which shows state-of-the-art performance on all of the cross-domain datasets investigated. The work is implemented using PyTorch [126, 127] and is publicly available[1]. For more details, please check the published paper [128].

4.1 Technical Approach

We first introduce our baseline model which simply extends image-base DA for videos using the temporal pooling mechanism (Section 4.1.1). And then we investigate better ways to incorporate temporal dynamics for video DA (Section 4.1.2), and describe our final proposed method with the domain attention mechanism (Section 4.1.3).

4.1.1 Baseline Model

Given the recent success of large-scale video classification using CNNs [11], we build our baseline on such architectures, as shown in the lower part of Figure 4.1.

[1] `https://github.com/cmhungsteve/TA3N`

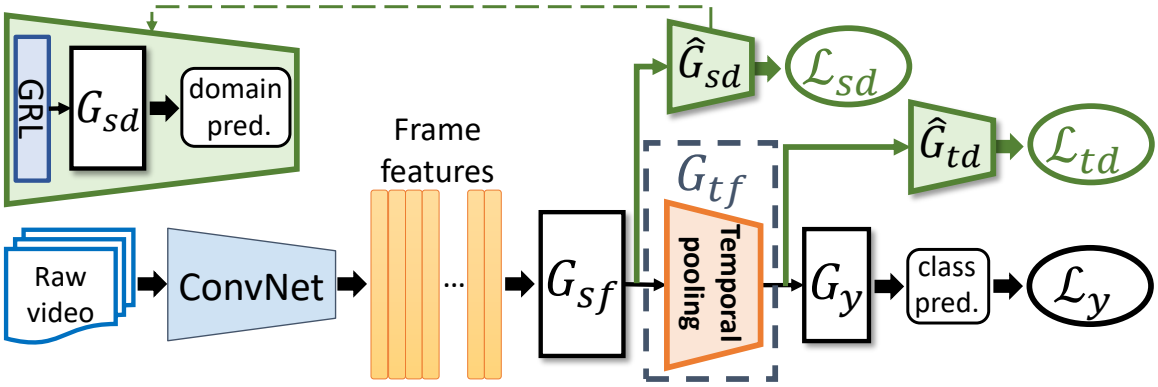


Figure 4.1: Baseline architecture (TemPooling) with the adversarial discriminators $\hat{G}_{sd}$ and $\hat{G}_{td}$.

We first feed the input video $X_i = \{x_i^1, x_i^2, ..., x_i^K\}$ extracted from ResNet [129] pre-trained on ImageNet into our model, where x_i^j is the jth frame-level feature representation of the ith video. The model can be divided into two parts: 1) *Spatial module* $G_{sf}(.;\theta_{sf})$, which consists of multilayer perceptrons (MLP) that aims to convert the general-purpose feature vectors into task-driven feature vectors, where the task is video classification in this paper; 2) *Temporal module* $G_{tf}(.;\theta_{tf})$ aggregates the frame-level feature vectors to form a single video-level feature vector for each video. In our baseline architecture, we conduct mean-pooling along the temporal direction to generate video-level feature vectors, and note it as *TemPooling*. Finally, another fully-connected layer $G_y(.;\theta_y)$ converts the video-level features into the final predictions, which are used to calculate the class prediction loss $\mathcal{L}_y$.

Similar to image-based DA problems, the baseline approach is not able to generalize to data from different domains due to domain shift. Therefore, we integrate TemPooling with the unsupervised DA method inspired by one of the most popular adversarial-based approaches, DANN [42, 43]. The main idea is to add additional domain classifiers $G_d(.;\theta_d)$, to discriminate whether the data is from the source or target domain. Before back-propagating the gradients to the main model, a gradient reversal layer (GRL) is inserted between G_d and the main model to invert the gradient, as shown in Figure 4.1. During adversarial training, the parameters θ_{sf} are learned by maximizing the domain discrimination loss $\mathcal{L}_d$, and parameters θ_d are learned by minimizing $\mathcal{L}_d$ with the domain label

d. Therefore, the feature generator G_f will be optimized to gradually align the feature distributions between the two domains.

In this paper, we note the *Adversarial Discriminator* $\hat{G}_d$ as the combination of a gradient reversal layer (GRL) and a domain classifier, and insert $\hat{G}_d$ into TemPooling in two ways: 1) $\hat{G}_{sd}$: show how directly applying image-based DA approaches can benefit video DA; 2) $\hat{G}_{td}$: indicate how DA on temporal-dynamics-encoded features benefits video DA.

The prediction loss $\mathcal{L}_y$, spatial domain loss $\mathcal{L}_{sd}$ and temporal domain loss $\mathcal{L}_{td}$ can be expressed as follows (ignoring all the parameter symbols through the paper to save space):

$$\mathcal{L}_y^i = L_y(G_y(G_{tf}(G_{sf}(X_i))), y_i) \tag{4.1}$$

$$\mathcal{L}_{sd}^i = \frac{1}{K}\sum_{j=1}^{K} L_d(G_{sd}(G_{sf}(x_i^j)), d_i) \tag{4.2}$$

$$\mathcal{L}_{td}^i = L_d(G_{td}(G_{tf}(G_{sf}(X_i))), d_i) \tag{4.3}$$

where K is the number of frames sampled from each video. L is the cross entropy loss function.

The overall loss can be expressed as follows:

$$\mathcal{L} = \frac{1}{N_S}\sum_{i=1}^{N_S}\mathcal{L}_y^i - \frac{1}{N_{S\cup T}}\sum_{i=1}^{N_{S\cup T}}(\lambda_s\mathcal{L}_{sd}^i + \lambda_t\mathcal{L}_{td}^i) \tag{4.4}$$

where N_S equals the number of source data, and $N_{S\cup T}$ equals the number of all data. λ_s and λ_t is the trade-off weighting for spatial and temporal domain loss.

4.1.2 Integration of Temporal Dynamics with DA

One main drawback of directly integrating image-based DA approaches into our baseline architecture is that the feature representations learned in the model are mainly from the spatial features. Although we implicitly encode the temporal information by the temporal pooling mechanism, the relation between frames is still missing. Therefore, we would like

to address two questions: 1) *Does the video DA problem benefit from encoding temporal dynamics into features?* 2) *Instead of only modifying feature encoding methods, how can DA be further integrated while encoding temporal dynamics into features?*

To answer the first question, given the fact that humans can recognize actions by reasoning the observations across time, we propose the *TemRelation* architecture by replacing the temporal pooling mechanism with the Temporal Relation module, which is modified from [130, 21], as shown in Figure 4.3.

The n-frame temporal relation is defined by the function:

$$R_n(V_i) = \sum_m g_{\phi^{(n)}}((V_i^n)_m) \tag{4.5}$$

where $(V_i^n)_m = \{v_i^a, v_i^b, ...\}_m$ is the mth set of frame-level representations from n temporal-ordered sampled frames. a and b are the frame indices. We fuse the feature vectors that are time-ordered with the function $g_{\phi^{(n)}}$, which is an MLP with parameters $\phi^{(n)}$. To capture temporal relations at multiple time scales, we sum up all the n-frame relation features into the final video representation. In this way, the temporal dynamics are explicitly encoded into features. We then insert $\hat{G}_d$ into TemRelation as we did for TemPooling.

Although aligning temporal-dynamic-encoded features benefits video DA, feature encoding and DA are still two separate processes, leading to sub-optimal DA performance. Therefore, we address the second question by proposing **Temporal Adversarial Adaptation Network (TA^{2}N)**, which explicitly integrates $\hat{G}_d$ *inside* the Temporal module to align the model across domains while learning temporal dynamics. Specifically, we integrate each n-frame relation with a corresponding relation discriminator $\hat{G}_{rd}^n$ because different n-frame relations represent different temporal characteristics, which correspond to different parts of actions. The relation domain loss $\mathcal{L}_{rd}$ can be expressed as follows:

$$\mathcal{L}_{rd}^i = \frac{1}{K-1} \sum_{n=2}^{K} L_d(G_{rd}^n(R_n(G_{sf}(X_i))), d_i) \tag{4.6}$$

The experimental results show that our integration strategy can effectively align domains spatio-temporally for videos, and outperform those which are extended from sophisticated

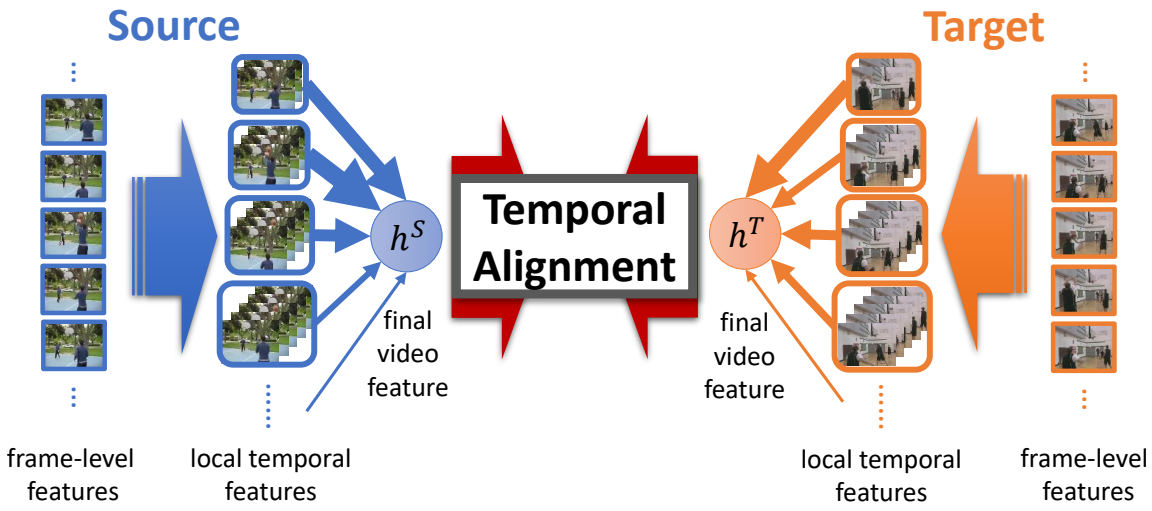


Figure 4.2: The domain attention mechanism in TA^{3}N. Thicker arrows correspond to larger attention weights.

DA approaches although TA^{2}N is adopted from a simpler DA method (DANN) (see details in Tables 4.5 to 4.7).

4.1.3 Temporal Attentive Alignment for Videos

The final video representation of TA^{2}N is generated by aggregating multiple local temporal features. Although aligning temporal features across domains benefits video DA, not all the features are equally important to align. In order to effectively align overall temporal dynamics, we want to focus more on aligning the local temporal features which have larger domain discrepancy. Therefore, we represent the final video representation as a combination of local temporal features with different attention weighting, as shown in Figure 4.2, and aim to attend to features of interest that are domain discriminative so that the DA mechanism can focus on aligning those features. The main question becomes: *How to incorporate domain discrepancy for attention?*

To address this, we propose **Temporal Attentive Adversarial Adaptation Network (TA^{3}N)**, as shown in Figure 4.3, by introducing the *domain attention* mechanism, which utilize the entropy criterion to generate the domain attention value for each n-frame relation

feature as below:

$$w_i^n = 1 - H(\hat{d}_i^n) \tag{4.7}$$

where $\hat{d}_i^n$ is the output of G_{rd}^n for the ith video. $H(p) = -\sum_k p_k \cdot \log(p_k)$ is the entropy function to measure uncertainty. w_i^n increases when $H(\hat{d}_i^n)$ decreases, which means the domains can be distinguished well. We also add a residual connection for more stable optimization. Therefore, the final video feature representation h_i generated from attended local temporal features, which are learned by local temporal modules $G_{tf}^{(n)}$, can be expressed as:

$$h_i = \sum_{n=2}^{K} (w_i^n + 1) \cdot G_{tf}^{(n)}(G_{sf}(X_i)) \tag{4.8}$$

Finally, we add the minimum entropy regularization to refine the classifier adaptation. However, we only want to minimize the entropy for the videos that are similar across domains. Therefore, we attend to the videos which have low domain discrepancy, so that we can focus more on minimizing the entropy for these videos. The attentive entropy loss $\mathcal{L}_{ae}$ can be expressed as follows:

$$\mathcal{L}_{ae}^i = (1 + H(\hat{d}_i)) \cdot H(\hat{y}_i) \tag{4.9}$$

where $\hat{d}_i$ and $\hat{y}_i$ is the output of G_{td} and G_y, respectively. We also adopt the residual connection for stability.

By combining Equations (4.1) to (4.3), (4.6) and (4.9), and replacing G_{sf} and G_{tf} with h_i by Equation (4.8), the overall loss of TA^{3}N can be expressed as follows:

$$\begin{aligned} \mathcal{L} = & \frac{1}{N_S} \sum_{i=1}^{N_S} \mathcal{L}_y^i + \frac{1}{N_{S \cup T}} \sum_{i=1}^{N_{S \cup T}} \gamma \mathcal{L}_{ae}^i \\ & - \frac{1}{N_{S \cup T}} \sum_{i=1}^{N_{S \cup T}} (\lambda^s \mathcal{L}_{sd}^i + \lambda^r \mathcal{L}_{rd}^i + \lambda^t \mathcal{L}_{td}^i) \end{aligned} \tag{4.10}$$

where λ^s, λ^r and λ^t is the trade-off weighting for each domain loss. γ is the weighting for the attentive entropy loss. All the weightings are chosen via grid search.

Our proposed TA^{3}N and TADA [93] both utilize entropy functions for attention but with different perspectives. TADA aims to focus on the foreground objects for image DA,

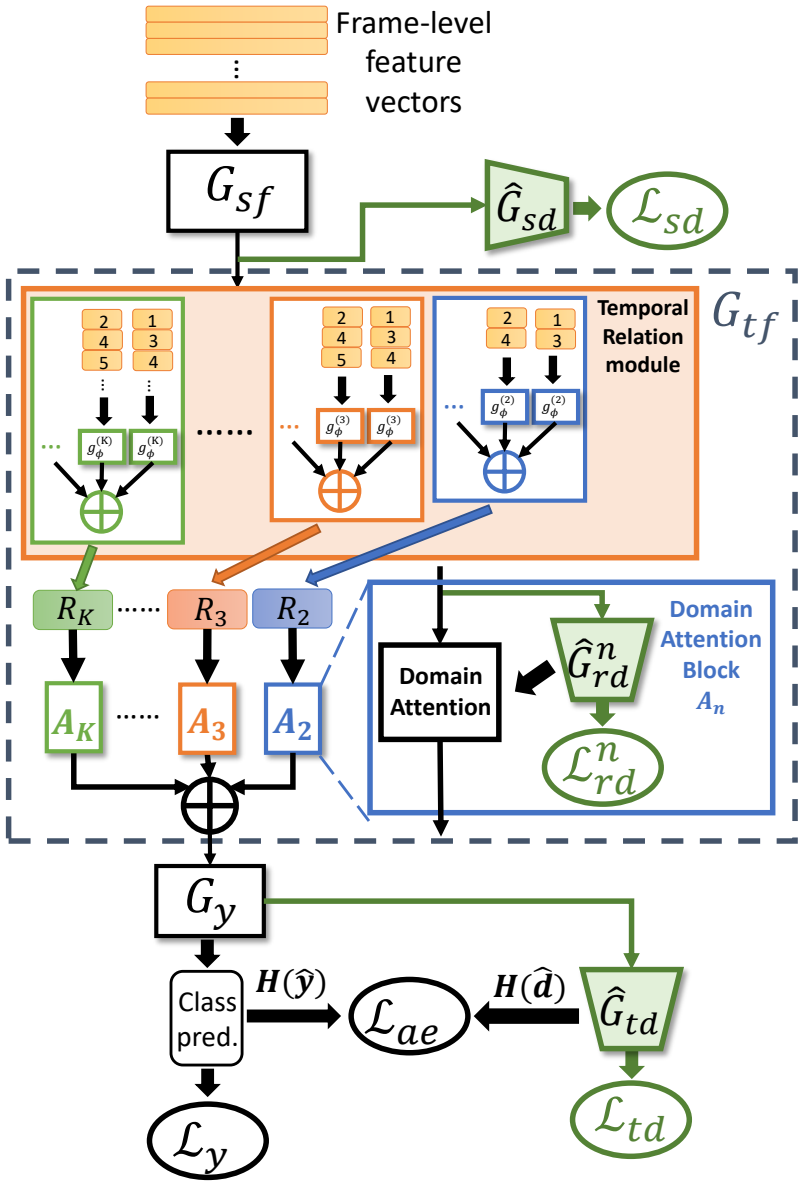


Figure 4.3: The overall architecture of the proposed Temporal Attentive Adversarial Adaptation Network (TA^{3}N) (rotated for clarity).

while TA^{3}N aims to find important and discriminative parts of temporal dynamics to align for video DA.

4.2 Datasets

There are very few benchmark datasets for video DA, and only small-scale datasets have been widely used [48, 49, 50]. Therefore, we specifically create two cross-domain datasets,

Table 4.1: The summary of the cross-domain video datasets.

	UCF-HMDB$_{small}$	UCF-Olympic	UCF-HMDB$_{full}$	Kinetics-Gameplay
length (sec.)	1 - 21	1 - 39	1 - 33	1 - 10
class #	5	6	12	30
resolution	UCF: 320×240 / Olympic: vary / HMDB: vary$\times 240$ Kinetics: vary / Gameplay: 1280×720			
frame rate	UCF: 25 / Olympic: 30 / HMDB: 30 Kinetics: vary / Gameplay: 30			
training video #	UCF: 482 HMDB: 350	UCF: 601 Olympic: 250	UCF: 1438 HMDB: 840	Kinetics: 43378 Gameplay: 2625
validation video #	UCF: 189 HMDB: 150	UCF: 240 Olympic: 54	UCF: 571 HMDB: 360	Kinetics: 3246 Gameplay: 749

UCF-HMDB$_{full}$ and **Kinetics-Gameplay**, to evaluate the proposed approaches for the video DA problem, as shown in Table 4.1.

4.2.1 UCF-HMDB$_{full}$

We extend UCF-HMDB$_{small}$ [48], which only selects 5 visually highly similar categories, by collecting all of the relevant and overlapping categories between UCF101 [23] and HMDB51 [22], which results in 12 categories: *climb, fencing, golf, kick_ball, pullup, punch, pushup, ride_bike, ride_horse, shoot_ball, shoot_bow,* and *walk*. Each category may correspond to multiple categories in the original UCF101 or HMDB51 dataset, as shown in Table 4.2. This dataset, **UCF-HMDB$_{full}$**, includes 1438 training videos and 571 validation videos from UCF, and 840 training videos and 360 validation videos from HMDB, as shown in Table 4.1. Most videos in UCF are from certain scenarios or similar environments, while videos in HMDB are in unconstrained environments and different camera angles, as shown in Figure 4.4. We follow the official split method to separate training and validation sets. This dataset, **UCF-HMDB$_{full}$**, includes more than 3000 video clips, which is around 3 times larger than UCF-HMDB$_{small}$ and UCF-Olympic.

Table 4.2: The lists of all collected categories in UCF and HMDB.

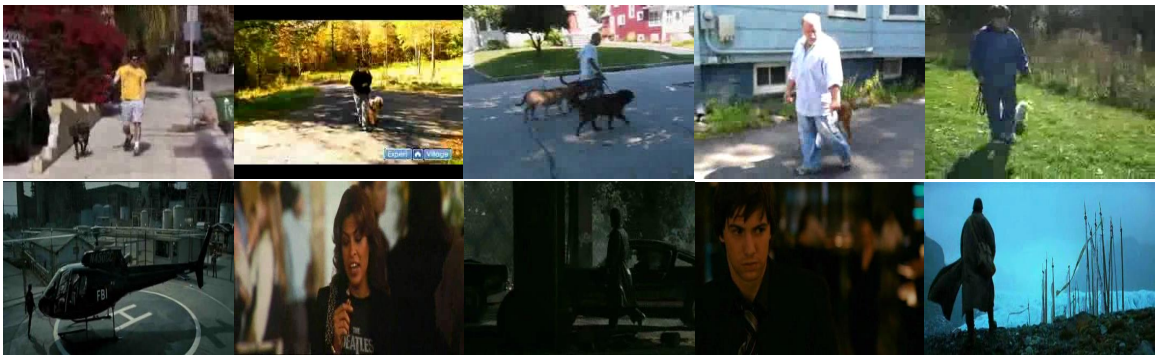
UCF-HMDB$_{full}$	UCF	HMDB
climb	RockClimbingIndoor, RopeClimbing	climb
fencing	Fencing	fencing
golf	GolfSwing	golf
kick_ball	SoccerPenalty	kick_ball
pullup	PullUps	pullup
punch	Punch, BoxingPunchingBag, BoxingSpeedBag	punch
pushup	PushUps	pushup
ride_bike	Biking	ride_bike
ride_horse	HorseRiding	ride_horse
shoot_ball	Basketball	shoot_ball
shoot_bow	Archery	shoot_bow
walk	WalkingWithDog	walk

4.2.2 Kinetics-Gameplay

In addition to real-world videos, we are also interested in virtual-world videos for DA. While there are more than ten real-world video datasets, there is a limited number of virtual-world datasets for video classification. It is mainly because rendering realistic human actions using game engines requires gaming graphics expertise which is time-consuming. Therefore, we create the *Gameplay* dataset by collecting gameplay videos from currently popular video games, *Detroit: Become Human* and *Fortnite*, to build our own video dataset for the virtual domain. The total length of the videos is 5 hours and 41 minutes. We segment all of the raw, untrimmed videos into video clips according to human annotations, which results in 91 categories: *argue, arrange_object, assemble_object, break, bump, carry, carve, chop_wood, clap, climb, close_door, close_others, crawl, cross_arm, crouch, crumple, cry, cut, dance, draw, drink, drive, eat, fall_down, fight, fix_hair, fly_helicopter, get_off, grab, haircut, hit, hit_break, hold, hug, juggle_coin, jump, kick, kiss, kneel, knock, lick, lie_down, lift, light_up, listen, make_bed, mop_floor, news_anchor, open_door, open_others, paint_brush, pass_object, pet, poke, pour, press, pull, punch, push, push_object, put_object,*

(a) fencing

(b) kick_ball



(c) walk

Figure 4.4: Snapshots of some example categories on UCF-HMDB$_{full}$.

raise_hand, read, row_boat, run, shake_hand, shiver, shoot_gun, sit, sit_down, slap, sleep, slide, smile, stand, stand_up, stare, strangle, swim, switch, take_off, talk, talk_phone, think, throw, touch, walk, wash_dishes, water_plant, wave_hand, and *weld*. The maximum length for each video clip is 10 seconds, and the minimum is 1 second. We also split the dataset into training, validation, and testing sets by randomly selecting videos in each category with the ratio 7:2:1.

For the real domain, we use one of the largest public video datasets *Kinetics-600* [8, 9]. We follow the closed-set DA setting [36] to select 30 overlapping categories between the Kinetics-600 and Gameplay datasets to build the **Kinetics-Gameplay** dataset with both domains: *break, carry, clean_floor, climb, crawl, crouch, cry, dance, drink, drive, fall_down, fight, hug, jump, kick, light_up, news_anchor, open door, paint_brush, paraglide, pour, push, read, run, shoot_gun, stare, talk, throw, walk,* and *wash_dishes*. Each category may also correspond to multiple categories in both datasets, as shown in Table 4.3. Kinetics-Gameplay includes 43378 training videos and 3246 validation videos from Kinetics, and 2625 training videos and 749 validation videos from Gameplay, as shown in Table 4.1. Kinetics-Gameplay is much more challenging than UCF-HMDB$_{full}$ due to the significant domain shift between the distributions of virtual and real data. Furthermore, The alignment between imbalanced-scaled source and target data is also another challenge. Some example snapshots are shown in Figure 4.5.

Figure 4.5: Some example screenshots from YouTube videos in Kinetics-Gameplay (left two: Gameplay, right two: Kinetics)

4.3 Experiments

We therefore evaluate DA approaches on four datasets: UCF-Olympic, UCF-HMDB$_{small}$, UCF-HMDB$_{full}$ and Kinetics-Gameplay.

Table 4.3: The lists of all collected categories in Kinetics and Gameplay.

Kinetics-Gameplay	Kinetics	Gameplay
break	breaking boards, smashing	break, bump, hit_break
carry	carrying baby	carry
clean_floor	mopping floor	mop_floor
climb	climbing a rope, climbing ladder, climbing tree, ice climbing, rock climbing	climb
crawl	crawling baby	crawl
crouch	squat, lunge	crouch, kneel
cry	crying	cry
dance	belly dancing, krumping, robot dancing	dance
drink	drinking shots, tasting beer	drink
drive	driving car, driving tractor	drive
fall_down	falling off bike, falling off chair, faceplanting	fall_down
fight	pillow fight, capoeira, wrestling, punching bag, punching person (boxing)	fight, strangle, punch, hit
hug	hugging (not baby), hugging baby	hug
jump	high jump, jumping into pool, parkour	jump
kick	drop kicking, side kick	kick
light_up	lighting fire	light_fire
news_anchor	news anchoring	news_anchor
open_door	opening door, opening refrigerator	open_door
paint_brush	brush painting	paint_brush
paraglide	paragliding	paraglide
pour	pouring beer	pour
push	pushing car, pushing cart, pushing wheelbarrow, pushing wheelchair, push up	push, push_object
read	reading book, reading newspaper	read
run	running on treadmill, jogging	run
shoot_gun	playing laser tag, playing paintball	shoot_gun
stare	staring	stare
talk	talking on cell phone, arguing, testifying	talk, argue, talk_phone
throw	throwing axe, throwing ball (not baseball or American football), throwing knife, throwing water balloon	throw
walk	walking the dog, walking through snow, jaywalking	walk
wash_dishes	washing dishes	wash_dishes

4.3.1 Experimental Setup

UCF-Olympic and UCF-HMDB$_{small}$: First, we evaluate our approaches on UCF-Olympic and UCF-HMDB$_{small}$, and compare with all other works that also evaluate on these two datasets [48, 49, 50]. We follow the default settings, but the method to split the UCF video clips into training and validations sets is not specified in these papers, so we follow the official split method from UCF101 [23].

UCF-HMDB$_{full}$ and Kinetics-Gameplay: For the self-collected datasets, we follow the common experimental protocol of unsupervised DA [36]: the training data consists of labeled data from the source domain and unlabeled data from the target domain, and the validation data is all from the target domain. However, unlike most of the image DA settings, our training and validation data in both domains are separate to avoid potentially overfitting while aligning different domains. To compare with image-based DA approaches, we extend several state-of-the-art methods [43, 41, 45, 46] for video DA with our TemPooling and TemRelation architectures, which are shown as follows:

1. *DANN [43]*: we add one adversarial discriminator $\hat{G}_{sd}$ right after the spatial module and add another one $\hat{G}_{td}$ right after the temporal module. We do not add one more discriminator for relation features for the fair comparison between TemPooling and TemRelation.

2. *JAN [41]*: we add Joint Maximum Mean Discrepancy (JMMD) to the final video representation and the class prediction.

3. *AdaBN [45]*: we integrate an adaptive batch-normalization layer into the feature generator G_{sf}. In the adaptive batch-normalization layer, the statistics (mean and variance) for both source and target domains are calculated, but only the target statistics are used for validating the target data.

4. *MCD [46]*: we add another classifier G'_y and follow the adversarial training proce-

dure of Maximum Classifier Discrepancy to iteratively optimize the generators (G_{sf} and G_{tf}) and the classifier (G_y).

The results are shown in Tables 4.5 to 4.7. The difference between the "Target only" and "Source only" settings is the domain used for training. The "Target only" setting can be regarded as the upper bound without domain shift while the "Source only" setting shows the lower bound which directly applies the model trained with source data to the target domain without modification.

4.3.2 Implementation Details

Detailed Architectures: The architecture with detailed notations for the baseline is shown in Figure 4.6. For our proposed TA^{3}N, after generating the n-frame relation features R_n by the temporal relation module, we calculate the domain attention value w^n using the domain prediction $\hat{d}$ from the relation discriminator G_{rd}^n, and then attend to R_n using w^n with a residual connection. To calculate the attentive entropy loss $\mathcal{L}_{ae}$, since the videos with low domain discrepancy are what we only want to focus on, we attend to the class entropy loss $H(\hat{y})$ using the domain entropy $H(\hat{d})$ as the attention value with a residual connection, as shown in Figure 4.7.

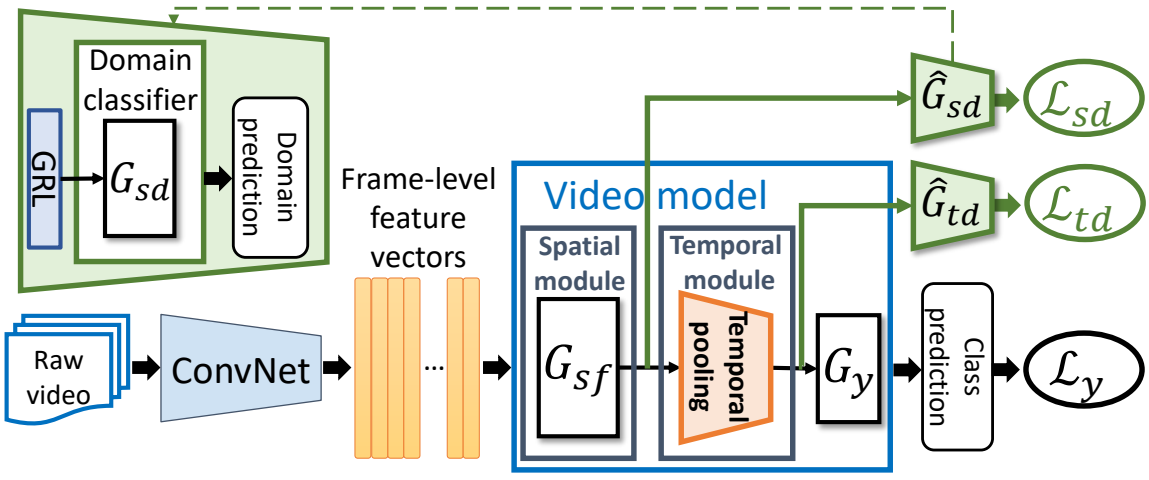


Figure 4.6: The detailed baseline architecture (TemPooling) with the adversarial discriminators $\hat{G}_{sd}$ and $\hat{G}_{td}$.

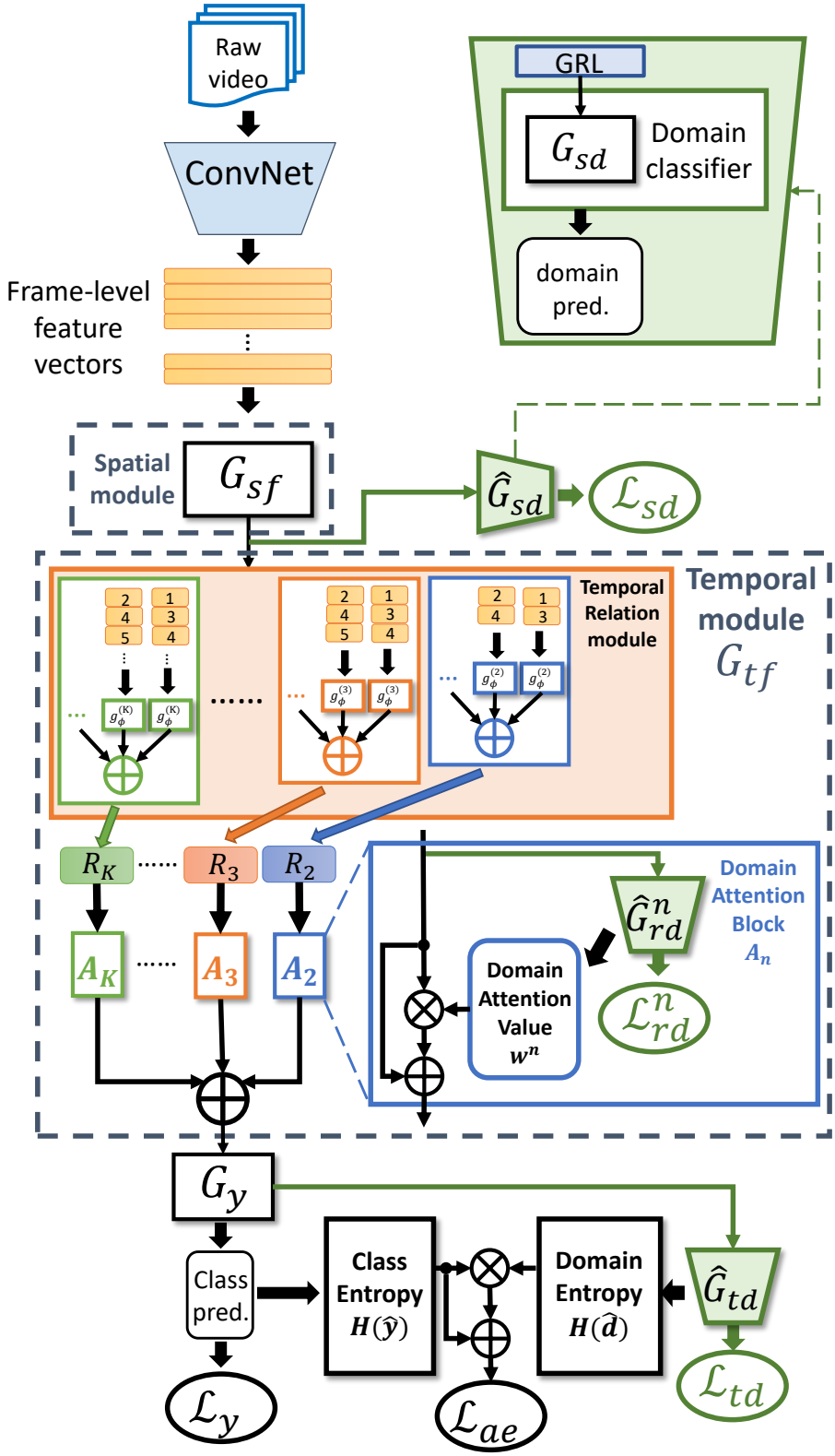


Figure 4.7: The detailed architecture of the proposed TA^{3}N (rotated for clarity).

Optimization: Our implementation is based on the PyTorch [126, 127] framework. We utilize the ResNet-101 model pre-trained on ImageNet as the frame-level feature extractor. We sample a fixed number K of frame-level feature vectors with equal spacing in the temporal direction for each video (K is equal to 5 in our setting to limit computational resource requirements). For optimization, the initial learning rate is 0.03, and we follow one of the commonly used learning-rate-decreasing strategies shown in DANN [43]. We use stochastic gradient descent (SGD) as the optimizer with the momentum and weight decay as 0.9 and 1×10^{-4}, respectively. The ratio between the source and target batch size is proportional to the scale between the source and target datasets. The source batch size depends on the scale of the dataset, which is 32 for UCF-Olympic and UCF-HMDB$_{small}$, 128 for UCF-HMDB$_{full}$ and 512 for Kinetics-Gameplay. The optimized values of λ^s, λ^r and λ^t are found using the coarse-to-fine grid-search approach. We first search using a coarse-grid with the geometric sequence [0, 10^{-3}, 10^{-2}, ..., 10^{0}, 10^{1}]. After finding the optimized range of values, [0, 1], we search again using a fine-grid with the arithmetic sequence [0, 0.25, ..., 1]. The final values are 0.75 for λ^s, 0.5 for λ^r and 0.75 for λ^t, respectively. We search γ only by a coarse-grid, and the best value is 0.3.

4.3.3 Experimental Results

UCF-Olympic and UCF-HMDB$_{small}$: In these two datasets, our approach outperforms all the previous methods by at least 6.5% absolute difference (98.15% - 91.60%) on the "U $\rightarrow$ O" setting, and 9% difference (99.33% - 90.25%) on the "U $\rightarrow$ H" setting, as shown in Table 4.4.

These results also show that the performance on these datasets is saturated. With a strong CNN as the backbone architecture, even our baseline architecture TemPooling can achieve high accuracy without any DA method (e.g. 96.3% for "U $\rightarrow$ O"). This suggests that these two datasets are not enough to evaluate more sophisticated DA approaches, so larger-scale datasets for video DA are needed.

Table 4.4: The accuracy (%) for the state-of-the-art work on UCF-Olympic and UCF-HMDB$_{small}$ (U: UCF, O: Olympic, H: HMDB).

Source → Target	U → O	O → U	U → H	H → U
W. Sultani et al. [48]	33.33	47.91	68.70	68.67
T. Xu et al. [49]	87.00	75.00	82.00	82.00
AMLS (GFK) [50]†	84.65	86.44	89.53	95.36
AMLS (SA) [50]†	83.92	86.07	90.25	94.40
DAAA [50]†‡	91.60	89.96	-	-
TemPooling	96.30	87.08	98.67	97.35
TemPooling + DANN [43]	**98.15**	90.00	**99.33**	98.41
Ours (TA^{2}N)	**98.15**	91.67	**99.33**	**99.47**
Ours (TA^{3}N)	**98.15**	**92.92**	**99.33**	**99.47**

UCF-HMDB$_{full}$: We then evaluate our approaches and compare with other image-based DA approaches on the UCF-HMDB$_{full}$ dataset, as shown in Tables 4.5 and 4.6. The accuracy difference between "Target only" and "Source only" indicates the *domain gap*. The gaps for the HMDB dataset are 11.11% for TemRelation and 10.28% for TemPooling (see Table 4.5), and the gaps for the UCF dataset are 21.01% for TemRelation and 17.16% for TemPooling (see Table 4.6). It is worth noting that the "Source only" accuracy of our baseline architecture (TemPooling) on UCF-HMDB$_{full}$ is much lower than UCF-HMDB$_{small}$ (e.g. 28.39 lower for "U → H"), which implies that UCF-HMDB$_{full}$ contains much larger domain discrepancy than UCF-HMDB$_{small}$. The value "Gain" is the difference from the "Source only" accuracy, which directly indicates the effectiveness of the DA approaches. We now answer the two questions for video DA in Section 4.1.2 (see Tables 4.5 and 4.6):

1. *Does the video DA problem benefit from encoding temporal dynamics into features?*

   From Tables 4.5 and 4.6, we see that for the same DA method, TemRelation outperforms TemPooling in most cases, especially for the gain value. For example, "TemPooling+DANN" reaches 0.83% absolute accuracy gain on the "U → H" setting and 0.17% gain on the "H → U" setting while "TemRelation+DANN" reaches 3.61% gain on "U → H" and 2.45% gain on "H → U". This means that applying DA approaches to the video representations which encode the temporal dynamics

Table 4.5: The comparison of accuracy (%) with other approaches on UCF-HMDB$_{full}$ (U $\rightarrow$ H).

Temporal Module	TemPooling		TemRelation	
	Acc.	Gain	Acc.	Gain
Target only	80.56	-	82.78	-
Source only	70.28	-	71.67	-
DANN [43]	71.11	0.83	75.28	3.61
JAN [41]	71.39	1.11	74.72	3.05
AdaBN [45]	75.56	5.28	72.22	0.55
MCD [46]	71.67	1.39	73.89	2.22
Ours (TA^{2}N)	N/A	-	77.22	5.55
Ours (TA^{3}N)	N/A	-	**78.33**	**6.66**

improves the overall performance for cross-domain video classification.

2. *How to further integrate DA while encoding temporal dynamics into features?*

   Although integrating TemRelation with image-based DA approaches generally has better alignment performance than the baseline (TemPooling), feature encoding and DA are still two separate processes. The alignment happens only before and after the temporal dynamics are encoded in features. In order to explicitly force alignment of the temporal dynamics across domains, we propose TA^{2}N, which reaches 77.22% (5.55% gain) on "U $\rightarrow$ H" and 80.56% (6.66% gain) on "H $\rightarrow$ U". Tables 4.5 and 4.6 show that although TA^{2}N is adopted from a simple DA method (DANN), it still outperforms other approaches which are extended from more sophisticated DA methods but do not follow our strategy.

Finally, with the domain attention mechanism, our proposed **TA^{3}N** reaches 78.33% (6.66% gain) on "U $\rightarrow$ H" and 81.79% (7.88% gain) on "H $\rightarrow$ U", achieving state-of-the-art performance on UCF-HMDB$_{full}$ in terms of accuracy and gain, as shown in Tables 4.5 and 4.6.

Kinetics-Gameplay: Kinetics-Gameplay is much more challenging than UCF-HMDB$_{full}$ because the data is from real and virtual domains, which have more severe domain shifts. Here we only utilize TemRelation as our backbone architecture since it is proved to out-

Table 4.6: The comparison of accuracy (%) with other approaches on UCF-HMDB$_{full}$ (H $\rightarrow$ U).

Temporal Module	TemPooling		TemRelation	
	Acc.	Gain	Acc.	Gain
Target only	92.12	-	94.92	-
Source only	74.96	-	73.91	-
DANN [43]	75.13	0.17	76.36	2.45
JAN [41]	80.04	5.08	79.69	5.79
AdaBN [45]	76.36	1.40	77.41	3.51
MCD [46]	76.18	1.23	79.34	5.44
Ours (TA^{2}N)	N/A	-	80.56	6.66
Ours (TA^{3}N)	N/A	-	**81.79**	**7.88**

perform TemPooling on UCF-HMDB$_{full}$. Table 4.7 shows that the accuracy gap between "Source only" and "Target only" is 47.27%, which is more than twice the number in UCF-HMDB$_{full}$. In this dataset, TA^{3}N also outperforms all the other DA approaches by increasing the "Source only " accuracy from 17.22% to 27.50%.

JAN [41] does not perform well on Kinetics-Gameplay compared to the performance on UCF-HMDB$_{full}$. The main reason is the imbalanced size between the source and target data in Kinetics-Gameplay. The discrepancy loss MMD is calculated using the same number of source and target data (not the case for other types of DA approaches). Therefore, in each iteration, MMD is calculated using parts of the source batch and the whole target batch. This means that the domain discrepancy is reduced only between part of source data and target data during training, so the learned model is still overfitted to the source domain. The discrepancy loss MMD works well when the source and target data are balanced, which is the case for most image DA datasets and UCF-HMDB$_{full}$, but not for Kinetics-Gameplay.

4.3.4 Ablation Study and Analysis

Integration of $\hat{G}_d$: We use UCF-HMDB$_{full}$ to investigate the performance for integrating $\hat{G}_d$ in different positions. There are three ways to insert the adversarial discriminator into our architectures, where each corresponds to different feature representations, leading to

Table 4.7: The comparison of accuracy (%) with other approaches on Kinetics-Gameplay.

	Acc.	Gain
Target only	64.49	-
Source only	17.22	-
DANN [43]	20.56	3.34
JAN [41]	18.16	0.94
AdaBN [45]	20.29	3.07
MCD [46]	19.76	2.54
Ours (TA^{2}N)	24.30	7.08
Ours (TA^{3}N)	**27.50**	**10.28**

Table 4.8: The full evaluation of accuracy (%) for integrating $\hat{G}_d$ in different positions without the attention mechanism.

S $\rightarrow$ T	UCF $\rightarrow$ HMDB		HMDB $\rightarrow$ UCF	
Temporal Module	TemPooling	TemRelation	TemPooling	TemRelation
Target only	80.56 (-)	82.78 (-)	92.12 (-)	94.92 (-)
Source only	70.28 (-)	71.67 (-)	74.96 (-)	73.91 (-)
$\hat{G}_{sd}$	71.11 (0.83)	74.44 (2.77)	75.13 (0.17)	74.44 (1.05)
$\hat{G}_{td}$	71.11 (0.83)	74.72 (3.05)	75.13 (0.17)	75.83 (1.93)
$\hat{G}_{rd}$	- (-)	76.11 (4.44)	- (-)	75.13 (1.23)
All $\hat{G}_d$	71.11 (0.83)	**77.22** (**5.55**)	75.13 (0.17)	**80.56** (**6.66**)

three types of discriminators $\hat{G}_{sd}$, $\hat{G}_{td}$ and $\hat{G}_{rd}$, which are shown in Figure 4.3 and the full experimental results are shown in Table 4.8. For the TemRelation architecture, the accuracy of utilizing $\hat{G}_{td}$ shows better performance than utilizing $\hat{G}_{sd}$ (averagely 0.58% absolute gain improvement across two tasks), while the accuracies are the same for TemPooling. This means that the temporal relation module can encode temporal dynamics that help the video DA problem, but temporal pooling cannot. Utilizing the relation discriminator $\hat{G}_{rd}$ can further improve the performance (0.92% improvement) since we simultaneously align and learn the temporal dynamics across domains. Finally, by combining all three discriminators, TA^{2}N improves even more (4.20% improvement).

Domain Attention Mechanism: We also apply the domain attention mechanism to TemPooling by attending to the raw frame features, as shown in Figure 4.8. Tables 4.9 and 4.10 show that the domain attention mechanism improves the performance for both TemPool-

ing and TemRelation architectures, including all types of adversarial discriminators. This implies that video DA can benefit from domain attention even if the backbone architecture does not encode temporal dynamics.

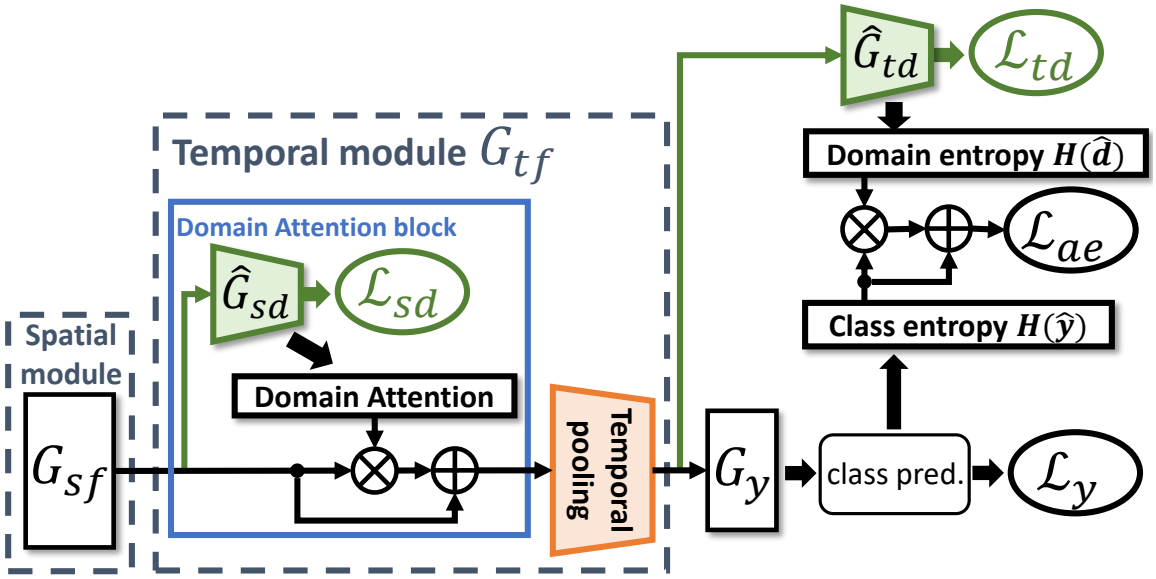


Figure 4.8: Baseline architecture (TemPooling) equipped with the domain attention mechanism.

Table 4.9: The evaluation of accuracy (%) for integrating $\hat{G}_d$ in different positions on "U $\rightarrow$ H".

Temporal Module	TemPooling	TemPooling + Attn.	TemRelation	TemRelation + Attn.
Target only	80.56 (-)		82.78 (-)	
Source only	70.28 (-)		71.67 (-)	
$\hat{G}_{sd}$	71.11 (0.83)	71.94 (1.66)	74.44 (2.77)	75.00 (3.33)
$\hat{G}_{td}$	71.11 (0.83)	72.78 (2.50)	74.72 (3.05)	76.94 (5.27)
$\hat{G}_{rd}$	- (-)	- (-)	76.11 (4.44)	76.94 (5.27)
All $\hat{G}_d$	71.11 (0.83)	**73.06** (**2.78**)	77.22 (5.55)	**78.33** (**6.66**)

We also compare the domain attention module with the general attention module, which calculates the attention weights via the *FC-Tanh-FC-Softmax* architecture. However, it performs worse since the weights are computed within one domain, lacking of the consideration of domain discrepancy, as shown in Table 4.11.

Table 4.10: The evaluation of accuracy (%) for integrating $\hat{G}_d$ in different positions on "H $\rightarrow$ U".

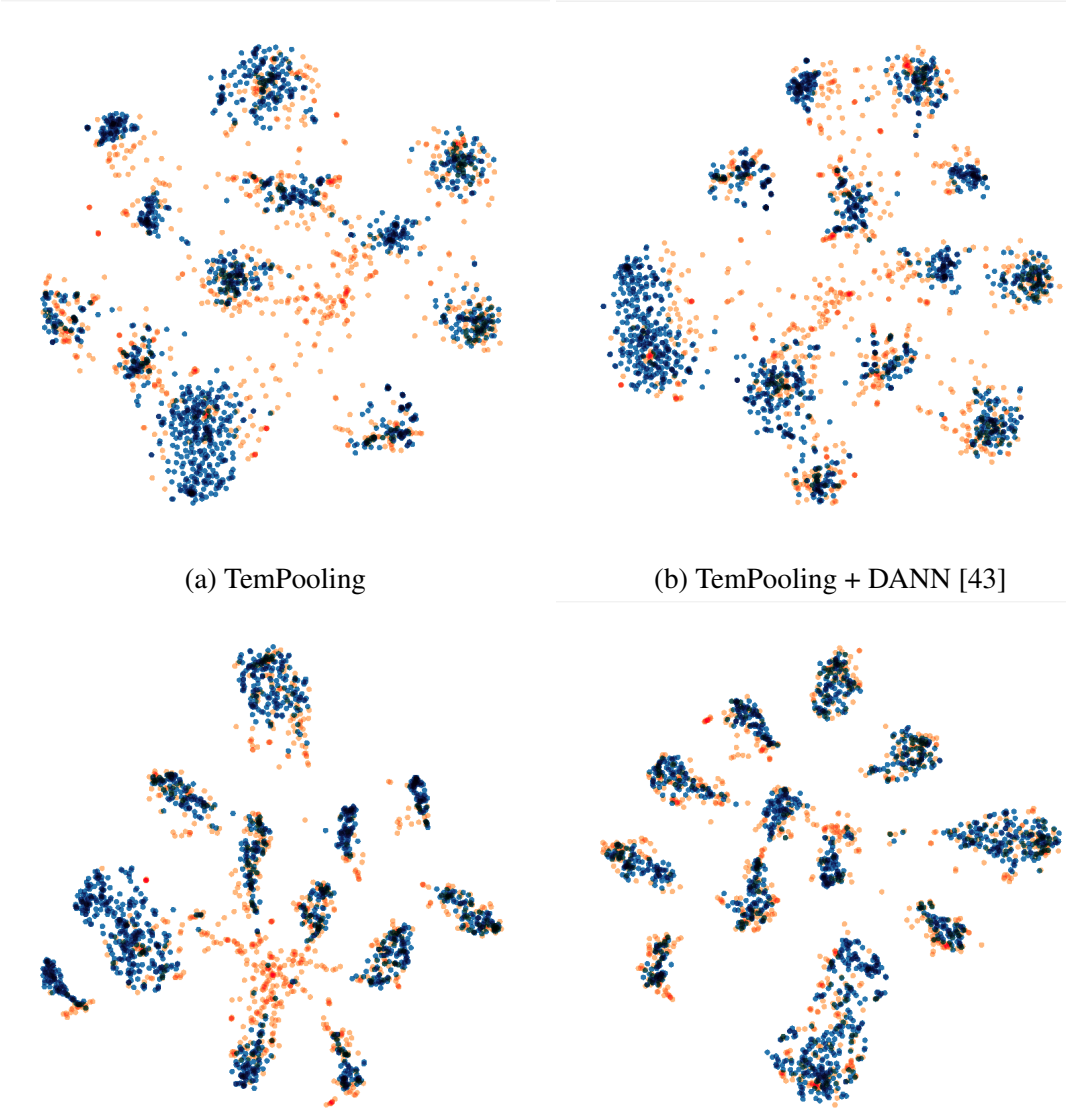
Temporal Module	TemPooling	TemPooling + Attn.	TemRelation	TemRelation + Attn.
Target only	92.12 (-)		94.92 (-)	
Source only	74.96 (-)		73.91 (-)	
$\hat{G}_{sd}$	75.13 (0.17)	77.58 (2.62)	74.44 (1.05)	78.63 (4.72)
$\hat{G}_{td}$	75.13 (0.17)	78.46 (3.50)	75.83 (1.93)	81.44 (7.53)
$\hat{G}_{rd}$	- (-)	- (-)	75.13 (1.23)	78.98 (5.07)
All $\hat{G}_d$	75.13 (0.17)	**78.46** (**3.50**)	80.56 (6.66)	**81.79** (**7.88**)

Table 4.11: The comparison of different attention methods.

S $\rightarrow$ T	UCF $\rightarrow$ HMDB	HMDB $\rightarrow$ UCF
Target only	82.78 (-)	94.92 (-)
Source only	71.67 (-)	73.91 (-)
No Attention	77.22 (5.55)	80.56 (6.66)
General Attention	77.22 (5.55)	80.91 (7.00)
Domain Attention	**78.33** (**6.66**)	**81.79** (**7.88**)

Visualization of Distribution: To investigate how our approaches bridge the gap between source and target domains, we visualize the distribution of both domains using t-SNE [131]. Figures 4.9a and 4.9b show that the models using the TemPooling architecture poorly align the distribution between different domains, even with the integration of image-based DA approaches. Figure 4.9c shows the temporal relation module helps to group source data (blue) into denser clusters but is still not able to generalize the distribution into the target domains (orange). Finally, with TA^{3}N, data from both domains are clustered and aligned with each other (Figure 4.9d).

Domain Discrepancy Measure: To measure the alignment between different domains, we use Maximum Mean Discrepancy (MMD) and domain loss, which are calculated using the final video representations. Lower MMD values and higher domain loss both imply smaller domain gap. TA^{3}N reaches lower discrepancy loss (0.0842) compared to the TemPooling baseline (0.184), and shows great improvement in terms of the domain loss (from 1.116 to 1.9286), as shown in Table 4.12.



(a) TemPooling (b) TemPooling + DANN [43]

(c) TemRelation (d) TA^{3}N

Figure 4.9: The comparison of t-SNE visualization with source (blue) and target (orange) distributions.

Table 4.12: The discrepancy loss (MMD), domain loss and validation accuracy of our baselines and proposed approaches.

	Discrepancy loss	Domain loss	Validation accuracy
TemPooling	0.1840	1.1163	70.28
TemPooling + DANN [43]	0.1604	1.2023	71.11
TemRelation	0.2626	1.7588	71.67
TA^{3}N	**0.0842**	**1.9286**	**78.33**

More Comparison:

1. *Other backbone architectures*: 3D ConvNets [14] have also been used for extracting video-level feature representations. However, 3D ConvNets consume a great deal of GPU memory, and [20] also shows that 3D ConvNets are limited by efficiency and effectiveness issues when extracting temporal information. Optical-flow extracts the motion characteristics between neighbor frames to compensate for the lack of temporal information in raw RGB frames. In this paper, we focus on attending to the temporal dynamics to effectively align domains even with only RGB frames. We consider optical-flow to be complementary to our method..

2. *Cycle-consistency*: Some papers related to cycle-consistency [132, 133] introduce self-supervised methods for learning visual correspondence between images or videos from unlabeled videos. They use cycle-consistency as free supervision to learn video representations. The main difference from our approach is that we explicitly align the feature spaces between source and target domains, while these self-supervised methods aim to learn general representations using only the source domain. We see cycle-consistency as a complementary method that can be integrated into our approach to achieve more effective domain alignment.

3. *Robotics*: In Robotics, it is a common trend to transfer the models trained in simulation to real world. One of the effective method to bridge the domain gap is randomizing the dynamics of the simulator during training to improve the robustness for different environments [134]. The setting is different from our task because we focus on feature learning rather than policy learning, and we see domain randomization as a complementary technique that can extend our approach to a more generalized version.

More Comparison:

1. *Testing time concern for TA^{3}N:* Different from TA^{2}N, TA^{3}N passes data to all the domain discriminators during testing. However, since all our domain discriminators are shallow, the testing time is similar. In our experiment, TA^{3}N only computes 10% more time than TA^{2}N.

2. *Failure cases for TemRelation:* Despite the significant overall improvement over TemPooling, TemRelation shows limited improvement for some categories with consistency across time. For example, with the same DA method (DANN), TemRelation has the same accuracy with TemPooling for *ride_bike* (97%), and has lower accuracy for *ride_horse* (93% and 97%). The possible reason is that temporal pooling can already model temporally consistent actions well, and it may be redundant to model these actions with multiple timescales like TemRelation.

CHAPTER 5

CROSS-DOMAIN VIDEO SEGMENTATION

Despite the recent progress of fully-supervised action segmentation techniques, the performance is still not fully satisfactory. One main challenge is the problem of spatio-temporal variations (e.g. different people may perform the same activity in various ways). Therefore, we exploit unlabeled videos to address this problem by reformulating the action segmentation task as a cross-domain problem with domain discrepancy caused by spatio-temporal variations. To reduce the discrepancy, we propose **Self-Supervised Temporal Domain Adaptation (SSTDA)**, which contains two self-supervised auxiliary tasks (binary and sequential domain prediction) to jointly align cross-domain feature spaces embedded with local and global temporal dynamics, achieving better performance than other Domain Adaptation (DA) approaches. On three challenging benchmark datasets (GTEA, 50Salads, and Breakfast), SSTDA outperforms the current state-of-the-art method by large margins, and requires much less labeled training data for comparable performance, demonstrating the usefulness of adapting to unlabeled target videos across variations. Figure 5.1 schematically illustrates the proposed framework for integrating SSTDA into the action segmentation pipeline. The work is implemented using PyTorch [126, 127] and is publicly available[1]. For more details, please check the published paper [135].

5.1 Technical Approach

In this section, the baseline model which is the current state-of-the-art for action segmentation, MS-TCN [33], is reviewed first (Section 5.1.1). Then the novel temporal domain adaptation scheme consisting of two self-supervised auxiliary tasks, binary domain prediction (Section 5.1.2) and sequential domain prediction (Section 5.1.2), is proposed, followed

[1] `https://github.com/cmhungsteve/SSTDA`

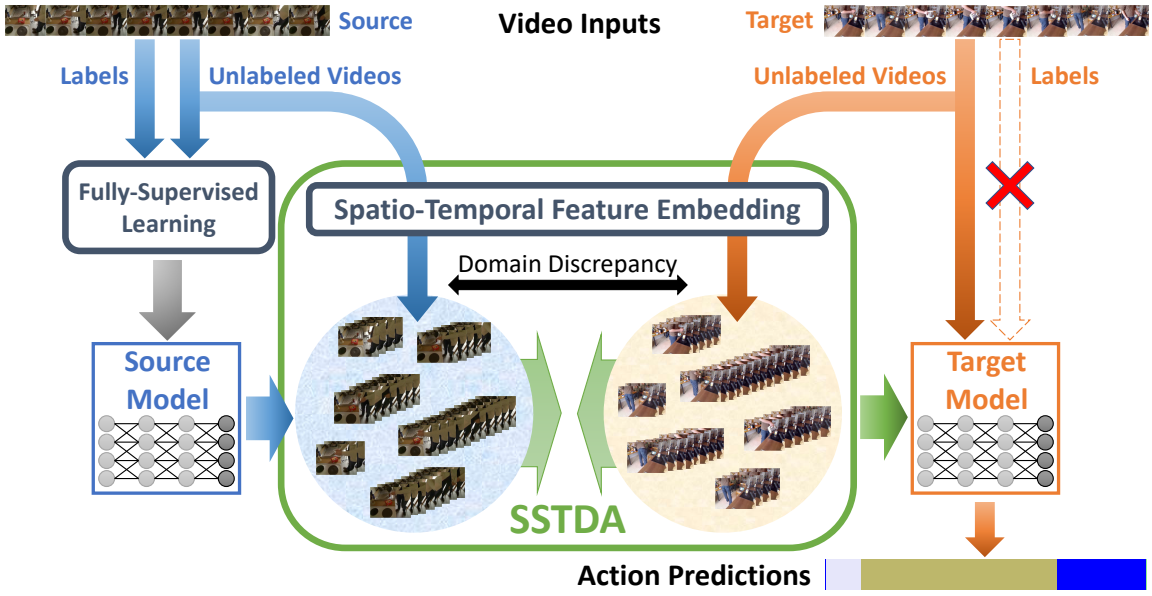


Figure 5.1: An overview of the proposed Self-Supervised Temporal Domain Adaptation (SSTDA) for action segmentation.

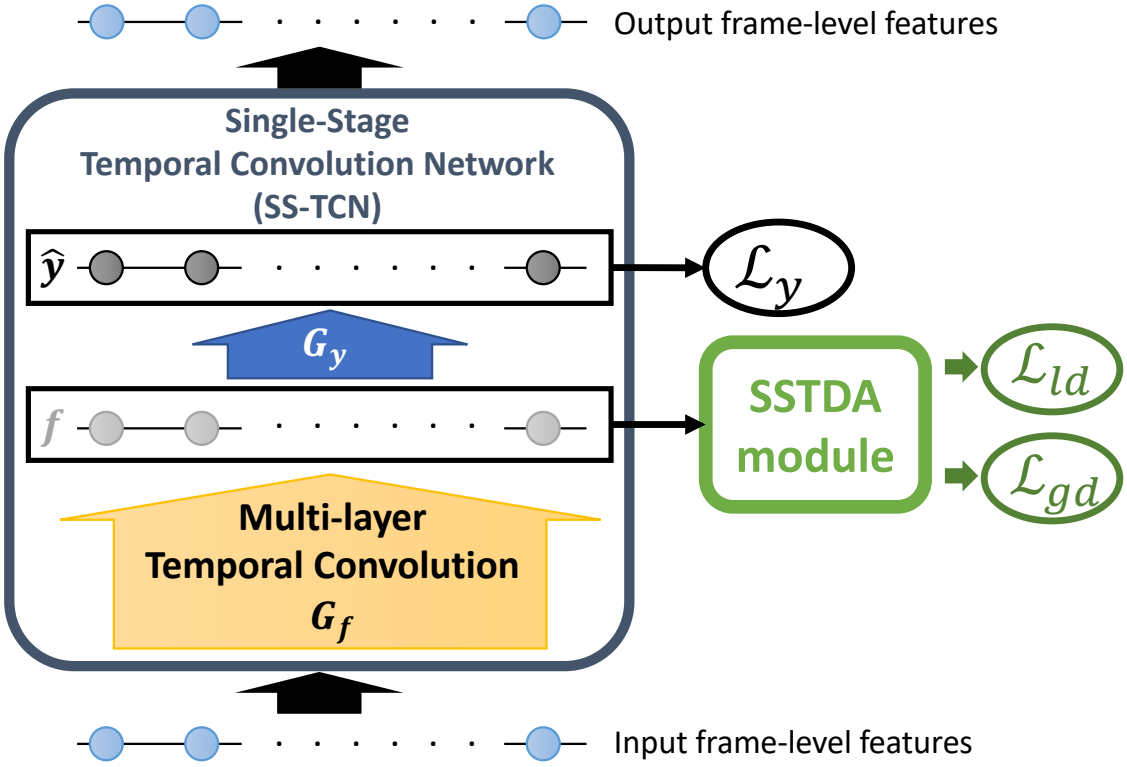


Figure 5.2: Illustration of the baseline model and the integration with our proposed SSTDA. Here we only show one stage in our multi-stage model.

by the final action segmentation model.

5.1.1 Baseline Model

Our work is built on the current state-of-the-art model for action segmentation, multi-stage temporal convolutional network (MS-TCN) [33]. For each stage, a single-stage TCN (SS-TCN) applies a multi-layer TCN, G_f, to derive the frame-level features $\boldsymbol{f} = \{f_1, f_2, ..., f_T\}$, and makes the corresponding predictions $\hat{\mathbf{y}} = \{\hat{y}_1, \hat{y}_2, ..., \hat{y}_T\}$ using a fully-connected layer G_y. By following [33], the prediction loss $\mathcal{L}_y$ is calculated based on the predictions $\hat{\mathbf{y}}$, as shown in the left part of Figure 5.2. Finally, multiple stages of SS-TCNs are stacked to enhance the temporal receptive fields, constructing the final baseline model, MS-TCN, where each stage takes the predictions from the previous stage as inputs, and makes predictions for the next stage.

The overall prediction loss function at each stage is designed as an integration of a classification loss and a smoothing loss [33] with the following form:

$$\mathcal{L}_y = \mathcal{L}_{cls} + \alpha \mathcal{L}_{T-MSE} \tag{5.1}$$

where $\mathcal{L}_{cls}$ is a standard cross-entropy loss, $\mathcal{L}_{T-MSE}$ is a truncated mean squared error for reduction of the discrepancy among neighboring frame-level predictions to enhance the smoothness, and α represents the trade-off coefficient for the smoothness loss. Minimization of the summary of the loss in Eq.5.1 on all stages will be derived for the whole model.

5.1.2 Self-Supervised Temporal Domain Adaptation

Despite the promising performance of MS-TCN on action segmentation over previous methods, there is still a large room for improvement. One main challenge is the problem of *spatio-temporal variations* of human actions [7], causing the distributional discrepancy across domains [38]. For example, different subjects may perform the same action completely differently due to personalized spatio-temporal styles. Moreover, collecting annotated data for action segmentation is challenging and time-consuming. Thus,

such challenges motivate the need to learn domain-invariant feature representations without full supervision. Inspired by the recent progress of self-supervised learning, which learns informative features that can be transferred to the main target tasks without external supervision (e.g. human annotation), we propose **Self-Supervised Temporal Domain Adaptation (SSTDA)** to diminish cross-domain discrepancy by designing self-supervised auxiliary tasks using unlabeled videos.

To effectively transfer knowledge, the self-supervised auxiliary tasks should be closely related to the main task, which is cross-domain action segmentation in this paper. Recently, adversarial-based DA approaches [42, 43] show progress in addressing cross-domain image problems using a domain discriminator with adversarial training where domain discrimination can be regarded as a self-supervised auxiliary task since domain labels are self-annotated. However, directly applying image-based DA for video tasks results in sub-optimal performance due to the temporal information being ignored [128]. Therefore, the question becomes: *How should we design the self-supervised auxiliary tasks to benefit cross-domain action segmentation?* More specifically, the answer should address both *cross-domain* and *action segmentation* problems.

To address this question, we first apply an auxiliary task *binary domain prediction* to predict the domain for each frame where the frame-level features are embedded with local temporal dynamics, aiming to address the cross-domain problems for videos in local scales. Then we propose a novel auxiliary task *sequential domain prediction* to temporally segment domains for untrimmed videos where the video-level features are embedded with global temporal dynamics, aiming to fully address the above question. Finally, SSTDA is achieved locally and globally by jointly applying these two auxiliary tasks, as illustrated in Figure 5.3.

In practice, since the key for effective video DA is to simultaneously align and learn temporal dynamics, instead of separating the two processes [128], we integrate SSTDA modules to multiple stages instead of the last stage only, and the single-stage integration is

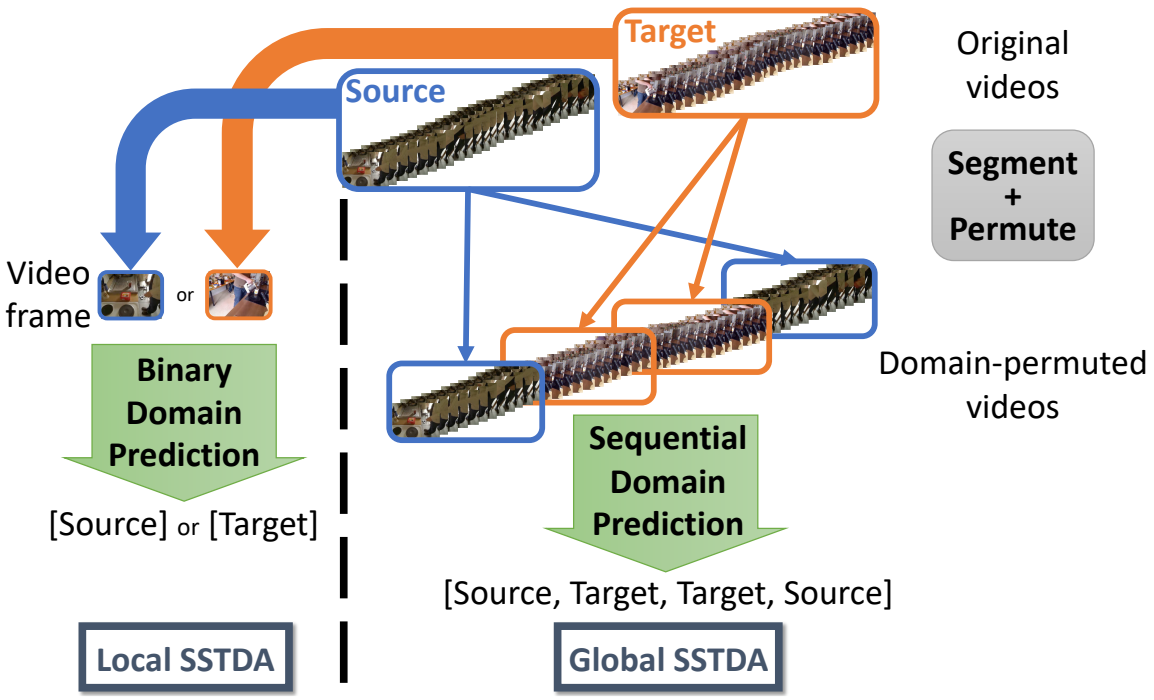


Figure 5.3: The two self-supervised auxiliary tasks in SSTDA: binary domain prediction and sequential domain prediction.

illustrated in Figure 5.2.

Local SSTDA: The main goal of action segmentation is to learn frame-level feature representations that encode spatio-temporal information so that the model can exploit information from multiple frames to predict the action for each frame. Therefore, we first learn domain-invariant frame-level features with the auxiliary task **Binary Domain Prediction** (Figure 5.3 left).

For a single stage, we feed the frame-level features from source and target domains $\boldsymbol{f^S}$ and $\boldsymbol{f^T}$, respectively, to an additional shallow *binary domain classifier* G_{ld}, to discriminate which domain the features come from. Since temporal convolution from previous layers encodes information from multiple adjacent frames to each frame-level feature, those frames contribute to the binary domain prediction for each frame. Through adversarial training with a gradient reversal layer (GRL) [42, 43], which reverses the gradient signs during back-propagation, G_f will be optimized to gradually align the feature distributions between the two domains. Here we note $\hat{G}_{ld}$ as G_{ld} equipped with GRL, as shown in

Figure 5.4.

Since this work is built on MS-TCN, *integrating $\hat{G}_{ld}$ with proper stages* is critical for effective DA. From our investigation, the best performance happens when $\hat{G}_{ld}$s are integrated into middle stages. See Section 5.2.4 for details.

The overall loss function becomes a combination of the baseline prediction loss $\mathcal{L}_y$ and the local domain loss $\mathcal{L}_{ld}$, which can be expressed as follows:

$$\mathcal{L} = \sum^{N_s} \mathcal{L}_y - \sum^{\widetilde{N_s}} \beta_l \mathcal{L}_{ld} \tag{5.2}$$

$$\mathcal{L}_{ld} = \frac{1}{T} \sum_{j=1}^{T} L_{ld}(G_{ld}(f_j), d_j) \tag{5.3}$$

where N_s is the total stage number in MS-TCN, $\widetilde{N_s}$ is the number of stages integrated with $\hat{G}_{ld}$, and T is the total frame number of a video. L_{ld} is a binary cross-entropy loss function, and β_l is the trade-off weight for local domain loss $\mathcal{L}_{ld}$, obtained by following the common strategy as [42, 43].

Global SSTDA: Although frame-level features $\boldsymbol{f}$ is learned using the context and dependencies from neighbor frames, the temporal receptive fields of $\boldsymbol{f}$ are still limited, unable to represent full videos. Solely integrating DA into $\boldsymbol{f}$ cannot fully address spatio-temporal variations for untrimmed long videos. Therefore, in addition to binary domain prediction for frame-level features, we propose the second self-supervised auxiliary task for video-level features: **Sequential Domain Prediction**, which predicts a sequence of domains for video clips, as shown in the right part of Figure 5.3. This task is a temporal domain segmentation problem, aiming to predict the correct permutation of domains for long videos consisting of shuffled video clips from both source and target domains. Since this goal is related to both cross-domain and action segmentation problems, *sequential domain prediction* can effectively benefit our main task.

More specifically, we first divide $\boldsymbol{f^S}$ and $\boldsymbol{f^T}$ into two sets of segments $\boldsymbol{F^S} = \{\boldsymbol{f_a^S}, \boldsymbol{f_b^S}, ...\}$ and $\boldsymbol{F^T} = \{\boldsymbol{f_a^T}, \boldsymbol{f_b^T}, ...\}$, respectively, and then learn the corresponding two sets of segment-

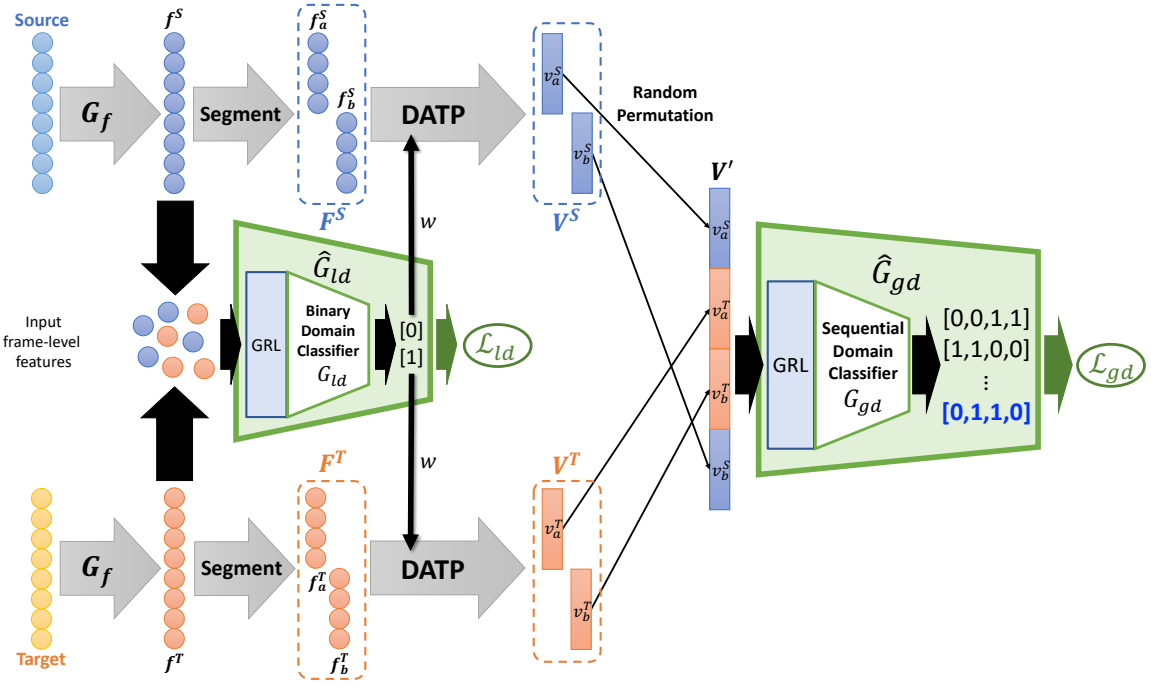


Figure 5.4: The overview of the proposed Self-Supervised Temporal Domain Adaptation (SSTDA).

level feature representations $\boldsymbol{V^S} = \{v_a^S, v_b^S, ...\}$ and $\boldsymbol{V^T} = \{v_a^T, v_b^T, ...\}$ with **Domain Attentive Temporal Pooling (DATP)**. All features v are then shuffled and combined in random order and fed to a *sequential domain classifier* G_{gd} equipped with GRL (noted as $\hat{G}_{gd}$) to predict the permutation of domains, as shown in Figure 5.4.

The details of two main modules in global SSTDA are shown as follows:

1. Domain Attentive Temporal Pooling (DATP): *Temporal pooling* is one of the most common methods to aggregate frame-level features into video-level features for each video. However, not all the frame-level features contribute the same to the overall domain discrepancy. Therefore, inspired by [128], we assign larger attention weights to the features which have larger domain discrepancy so that we can focus more on aligning those features, achieving more effective domain adaptation.

   More specifically, we utilize the entropy criterion to generate the domain attention

value for each frame-level feature f_j as below:

$$\hat{w}_j = 1 - H(\hat{d}_j) \tag{5.4}$$

where $\hat{d}_j$ is the domain prediction from $\hat{G}_{ld}$. $H(p) = -\sum_k p_k \cdot \log(p_k)$ is the entropy function to measure uncertainty. $\hat{w}_j$ increases when $H(\hat{d}_j)$ decreases, which means the domains can be distinguished well. We also add a residual connection for more stable optimization. Finally, we aggregate the attended frame-level features with temporal pooling to generate the video-level feature v, which is noted as *Domain Attentive Temporal Pooling (DATP)*, as illustrated in the left part of Figure 5.5 and can be expressed as:

$$v = \frac{1}{T'}\sum_{j=1}^{T'}(\hat{w}_j + 1) \cdot f_j = \frac{1}{T'}\sum_{j=1}^{T'} w_j \cdot f_j \tag{5.5}$$

where $+1$ refers to the residual connection, and the attention weights w_j equal to $\hat{w}_j + 1$. T' is the number of frames used to generate a video-level feature.

2. Sequential Domain Prediction: By separately applying DATP to both source and target segments, respectively, a set of segment-level feature representations $\boldsymbol{V} = \{v_a^S, v_b^S, ..., v_a^T, v_b^T, ...\}$ are obtained. We then shuffle all the features in $\boldsymbol{V}$ and concatenate them into a feature to represent a long and untrimmed video $\boldsymbol{V'}$, which contains video segments from both domains in random order. Finally, $\boldsymbol{V'}$ is fed into a *sequential domain classifier* G_{gd} to predict the permutation of domains for the video segments. For example, if $\boldsymbol{V'} = [v_a^S, v_a^T, v_b^T, v_b^S]$, the goal of G_{gd} is to predict the permutation as $[0, 1, 1, 0]$. G_{gd} is a multi-class classifier where the class number corresponds to the total number of all possible permutations of domains, and the complexity of G_{gd} is determined by the segment number for each video (more analyses in Section 5.2.4). The outputs of G_{gd} are used to calculate the global domain loss $\mathcal{L}_{gd}$ as below:

$$\mathcal{L}_{gd} = L_{gd}(G_{gd}(\boldsymbol{V'})), y_d) \tag{5.6}$$

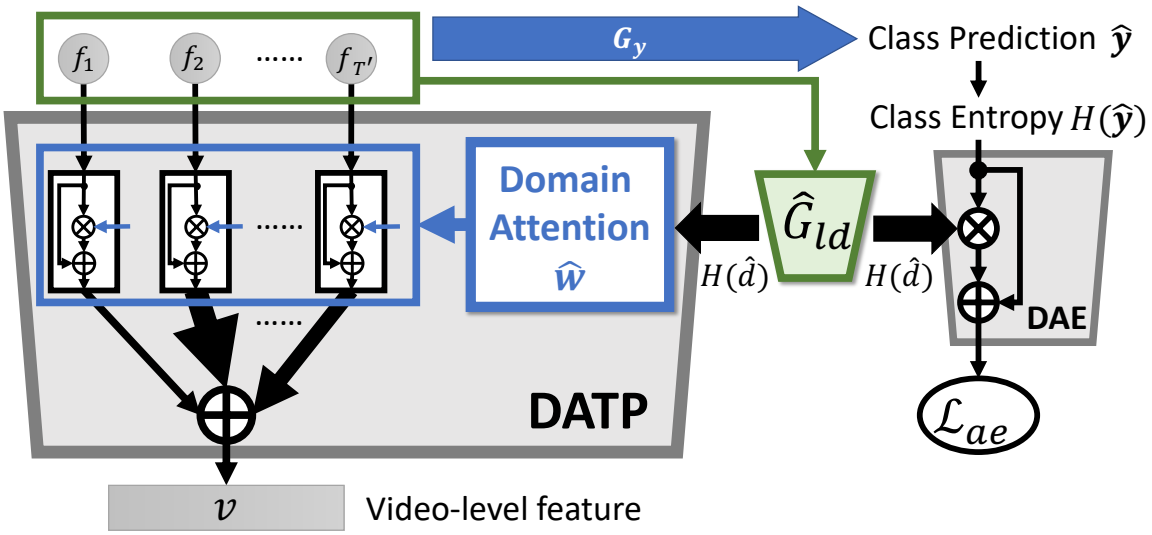


Figure 5.5: The details of DATP (*left*) and DAE (*right*).

where L_{gd} is also a standard cross-entropy loss function where the class number is determined by the segment number. Through adversarial training with GRL, *sequential domain prediction* also contributes to optimizing G_f to align the feature distributions between the two domains.

There are some self-supervised learning works also proposing the concepts of *temporal shuffling* [98, 102]. However, they predict temporal orders within one domain, aiming to learn general temporal information for video features. Instead, our method predicts temporal permutation for cross-domain videos, which are shown with a dual-branch pipeline in Figure 5.4, and integrate with binary domain prediction to effectively address both *cross-domain* and *action segmentation* problems.

5.1.3 Full Architecture

In addition to the local and global domain losses, we also add a domain attentive entropy (DAE) loss $\mathcal{L}_{ae}$ as follows: **Domain Attentive Entropy (DAE):** Minimum entropy regularization is a common strategy to perform more refined classifier adaptation. However, we only want to minimize class entropy for the frames that are similar across domains. Therefore, inspired by [93], we attend to the frames which have low domain discrepancy,

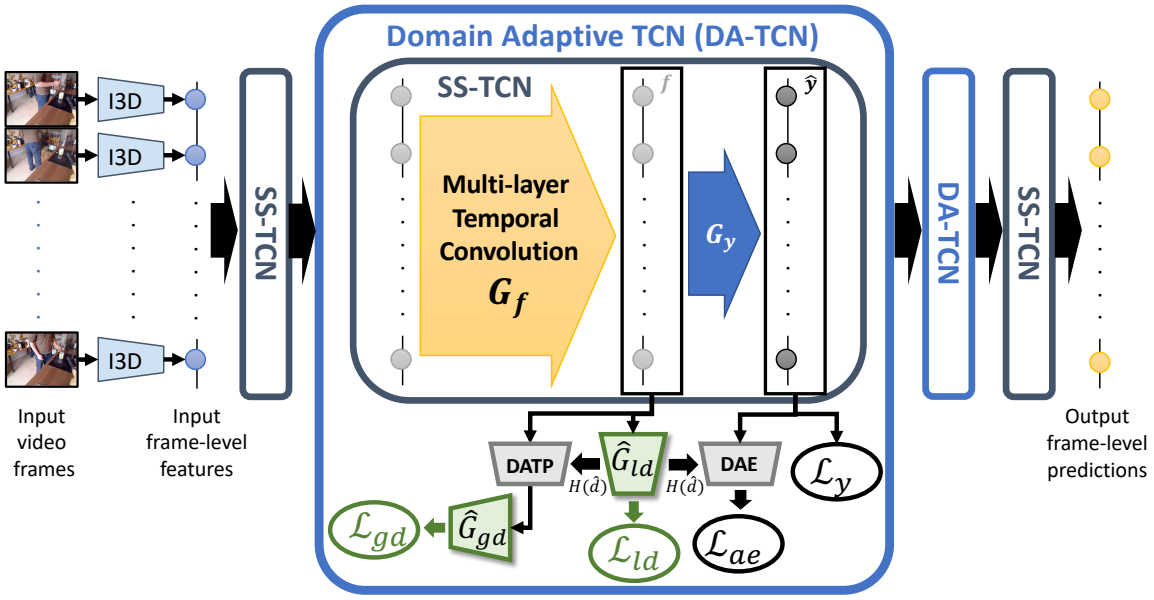


Figure 5.6: The full architecture of the proposed SSTDA with the action segmentation pipeline.

corresponding to high domain entropy $H(\hat{d}_j)$. More specifically, we adopt the *Domain Attentive Entropy (DAE)* module to calculate the attentive entropy loss $\mathcal{L}_{ae}$, which can be expressed as follows:

$$\mathcal{L}_{ae} = \frac{1}{T}\sum_{j=1}^{T}(H(\hat{d}_j)+1)\cdot H(\hat{y}_j) \tag{5.7}$$

where $\hat{d}$ and $\hat{y}$ is the output of $\hat{G}_{ld}$ and G_y, respectively. T is the total frame number of a video. We also apply the residual connection for stability, as shown in the right part of Figure 5.5.

Our method is built upon the state-of-the-art action segmentation model, MS-TCN [33], which takes input frame-level feature representations and generates the corresponding output frame-level class predictions by four stages of SS-TCN. In our implementation, we convert the second and third stages into *Domain Adaptive TCN (DA-TCN)* by integrating each SS-TCN with the following three parts: 1) $\hat{G}_{ld}$ (for *binary domain prediction*), 2) DATP and $\hat{G}_{gd}$ (for *sequential domain prediction*), and 3) DAE, bringing three corresponding loss functions, $\mathcal{L}_{ld}$, $\mathcal{L}_{gd}$ and $\mathcal{L}_{ae}$, respectively, as illustrated in Figure 5.6. The overall loss function of our final proposed **Self-Supervised Temporal Domain Adaptation (SSTDA)** can

Table 5.1: The statistics of action segmentation datasets.

	GTEA	**50Salads**	**Breakfast**
subject #	4	25	52
class #	11	17	48
video #	28	50	1712
avg. length (min.)	1	6.4	2.7
avg. action #/video	20	20	6
cross-validation	4-fold	5-fold	4-fold
leave-#-subject-out	1	5	13

be formulated as below:

$$\mathcal{L} = \sum^{N_s} \mathcal{L}_y - \sum^{\widetilde{N_s}} (\beta_l \mathcal{L}_{ld} + \beta_g \mathcal{L}_{gd} - \mu \mathcal{L}_{ae}) \tag{5.8}$$

where β_l, β_g and μ are the weights for $\mathcal{L}_{ld}$, $\mathcal{L}_{gd}$ and $\mathcal{L}_{ae}$, respectively, obtained by the methods described in Section 5.2.2. s is the stage index in MS-TCN. N_s is the total stage number, while $\widetilde{N_s}$ is the stage number of DA-TCN.

5.2 Experiments

To validate the effectiveness of the proposed methods in reducing spatial-temporal discrepancy for action segmentation, we choose three challenging datasets: GTEA [53], 50Salads [54], and Breakfast [55], which separate the training and testing sets by different people (noted as *subjects*), resulting in high spatio-temporal variations.

5.2.1 Datasets and Evaluation Metrics

The three evaluated datasets are shown as follows:

1. The **GTEA** dataset has 28 videos with 7 kitchen activities (e.g. *making coffee*) performed by 4 subjects. It contains 11 actions including background, and each video has 20 action instances with the length of around one minute. We apply 4-fold cross-validation by leaving one subject out in evaluation.

2. The **50Salads** dataset includes 50 videos for salad preparation activities performed by 25 subjects. There are 17 action classes in total. Each video averagely contains 20 action instances with an average length of 6.4 minutes. For evaluation, we utilize 5-fold cross-validation by leaving five subjects out.

3. The **Breakfast** dataset is the largest dataset with 1712 videos for breakfast preparation activities done by 52 subjects. All videos were taken in 18 different kitchens with 48 action classes where each video contains 6 action instances on average and is around 2.7 minutes long. For evaluation, we use the standard 4-fold cross-validation by leaving 13 subjects out.

The overall statistics and the evaluation protocols of the three datasets are listed in Table 5.1. We follow [30] to use the following three metrics for evaluation:

1. **Frame-wise accuracy (Acc)**: Acc is one of the most typical evaluation metrics for action segmentation, but it does not consider the temporal dependencies of the prediction, causing the inconsistency between qualitative assessment and frame-wise accuracy. Besides, long action classes have higher impact on this metric than shorter action classes, making it not able to reflect over-segmentation errors.

2. **Segmental edit score (Edit)**: The edit score penalizes over-segmentation errors by measuring the ordering of predicted action segments independent of slight temporal shifts.

3. **Segmental F1 score at the IoU threshold $k\%$ (F1@k)**: F1@k also penalizes over-segmentation errors while ignoring minor temporal shifts between the predictions and ground truth. The scores are determined by the total number of actions but do not depend on the duration of each action instance, which is similar to mean average precision (mAP) with intersection-over-union (IoU) overlap criteria. F1@k becomes popular recently since it better reflects the qualitative results.

5.2.2 Implementation and Optimization

Our implementation is based on the PyTorch [126, 127] framework. We extract I3D [18] features for the video frames and use these features as inputs to our model. The video frame rates are the same as [33]. For GTEA and Breakfast datasets we use a video temporal resolution of 15 frames per second (fps), while for 50Salads we downsampled the features from 30 fps to 15 fps to be consistent with the other datasets. For fair comparison, we adopt the same architecture design choices of MS-TCN [33] as our baseline model. The whole model consists of four stages where each stage contains ten dilated convolution layers. We set the number of filters to 64 in all the layers of the model and the filter size is 3. For optimization, we utilize the Adam optimizer and a batch size equal to 1, following the official implementation of MS-TCN [33]. Since the target data size is smaller than the source data, each target data is loaded randomly multiple times in each epoch during training. For the weighting of loss functions, we follow the common strategy as [42, 43] to gradually increase β_l and β_g from 0 to 1. The weighting α for smoothness loss is 0.15 as in [33] and μ is chosen as 1×10^{-2} via the grid-search.

5.2.3 Experimental Results

We first investigate the effectiveness of our approaches in utilizing unlabeled target videos for action segmentation. "Source only" means the model is trained only with source labeled videos, i.e., MS-TCN [33]. And then our approach is compared to other DA methods with the same setting. Finally, we compare the proposed approach to the state-of-the-art methods on all three datasets, and investigate how our method can reduce the reliance on source labeled data.

Self-Supervised Temporal Domain Adaptation: First we investigate the performance of local SSTDA by integrating the auxiliary task *binary domain prediction* with the baseline model. The results on all three datasets with respect to all the metrics are improved significantly, as shown in Table 5.2. For example, on the GTEA dataset, our approach

Table 5.2: The experimental results for our approaches on three benchmark datasets.

GTEA	F1@{10, 25, 50}			Edit	Acc
Source only (MS-TCN)†	86.5	83.6	71.9	81.3	76.5
Local SSTDA	89.6	87.9	74.4	84.5	**80.1**
SSTDA‡	**90.0**	**89.1**	**78.0**	**86.2**	79.8
50Salads	F1@{10, 25, 50}			Edit	Acc
Source only (MS-TCN)†	75.4	73.4	65.2	68.9	82.1
Local SSTDA	79.2	77.8	70.3	72.0	82.8
SSTDA‡	**83.0**	**81.5**	**73.8**	**75.8**	**83.2**
Breakfast	F1@{10, 25, 50}			Edit	Acc
Source only (MS-TCN)†	65.3	59.6	47.2	65.7	64.7
Local SSTDA	72.8	67.8	55.1	71.7	**70.3**
SSTDA‡	**75.0**	**69.1**	**55.2**	**73.7**	70.2

outperforms the baseline by 4.3% for F1@25, 3.2% for the edit score and 3.6% for the frame-wise accuracy. Although local SSTDA mainly works on the frame-level features, the temporal information is still encoded using the context from neighbor frames, helping address the variation problem for videos across domains.

Despite the improvement from local SSTDA, integrating DA into frame-level features cannot fully address the problem of spatio-temporal variations for long videos. Therefore, we integrate our second proposed auxiliary task sequential domain prediction for untrimmed long videos. By jointly training with both auxiliary tasks, SSTDA can jointly align cross-domain feature spaces embedding with local and global temporal dynamics, and further improve over local SSTDA with significant margins. For example, on the 50Salads dataset, it outperforms local SSTDA by 3.8% for F1@10, 3.7% for F1@25, 3.5% for F1@50, and 3.8% for the edit score, as shown in Table 5.2.

One interesting finding is that local SSTDA contributes to most of the frame-wise accuracy improvement for SSTDA because it focuses on aligning frame-level feature spaces. On the other hand, sequential domain prediction benefits aligning video-level feature spaces, contributing to further improvement for the other two metrics, which consider temporal relation for evaluation.

Learning from Unlabeled Target Videos: We first compare SSTDA with other pop-

ular DA approaches [43, 41, 136, 137, 46, 92] to validate the effectiveness of reducing spatio-temporal discrepancy with the same amount of unlabeled target videos. Table 5.3 shows that our proposed SSTDA outperforms all the other investigated DA methods in terms of the two metrics that consider temporal relation. We conjecture the main reason is that all these DA approaches are designed for cross-domain image problems. Although they are integrated with frame-level features which encode local temporal dynamics, the limited temporal receptive fields prevent them from fully addressing temporal domain discrepancy. Instead, the *sequential domain prediction* in SSTDA is directly applied to the whole untrimmed video, helping to globally align the cross-domain feature spaces that embed longer temporal dynamics, so that spatio-temporal variations can be reduced more effectively.

We also compare with the most recent video-based self-supervised learning method, [102], which can also learn temporal dynamics from unlabeled target videos. However, the performance is even worse than other DA methods, implying that temporal shuffling *within single domain* does not effectively benefit cross-domain action segmentation, resulting in worse performance than other DA methods. Instead, our proposed self-supervised auxiliary tasks make predictions on cross-domain data, achieving significant improvement in the performance of our main task, action segmentation.

For the fair comparison, we integrate all these methods with the same baseline model, where the single-stage integration methods are described as follows:

1. *DANN [43]*: We add one discriminator, which is the same as G_{ld}, equipped a gradient reversal layer (GRL) to the final frame-level features $\boldsymbol{f}$.

2. *JAN [41]*: We integrate Joint Maximum Mean Discrepancy (JMMD) to the final frame-level features $\boldsymbol{f}$ and the class prediction $\hat{\boldsymbol{y}}$.

3. *MADA [136]*: Instead of a single discriminator, we add multiple discriminators according to the class number to calculate the domain loss for each class. All the class-

based domain losses are weighted with prediction probabilities and then summed up to obtain the final domain loss.

4. *MSTN [137]*: We utilize pseudo-labels to cluster the data from the source and target domains, and calculate the class centroids for the source and target domain separately. Then we compute the semantic loss by calculating mean squared error (MSE) between the source and target centroids. The final loss contains the prediction loss, the semantic loss, and the domain loss as DANN [43].

5. *MCD [46]*: We apply another classifier G'_y and follow the adversarial training procedure of Maximum Classifier Discrepancy to iteratively optimize the generator (G_f in our case) and the classifier (G_y). The L1-distance is used as the discrepancy loss.

6. *SWD [92]*: The framework is similar to MCD, but we replace the L1-distance with the Wasserstein distance as the discrepancy loss.

7. *VCOP [102]*: We divide $\boldsymbol{f}$ into three segments and compute the segment-level features with temporal pooling. After temporal shuffling the segment-level features, pairwise features are computed and concatenated into the final feature representing the video clip order. The final features are then fed into a shallow classifier to predict the order.

.

Less Training Labeled Data: Given the significant improvement by exploiting unlabeled target videos, it implies the potential to train with fewer number of labeled frames using SSTDA. In this setting, we drop labeled frames from source domains with uniform sampling for training, and evaluate on the same length of validation data. Our experiment on the 50Salads dataset shows that by integrating with SSTDA, the performance does not drop significantly with the decrease in labeled training data, indicating the alleviation of reliance on labeled training data. Finally, only *65%* of labeled training data are required to achieve

Table 5.3: The comparison of different methods that can learn information from unlabeled target videos.

GTEA	F1@{10, 25, 50}			Edit
Source only (MS-TCN)	86.5	83.6	71.9	81.3
VCOP [102]	87.3	85.9	70.1	82.2
DANN [43]	89.6	87.9	74.4	84.5
JAN [41]	88.7	87.6	73.1	83.1
MADA [136]	88.6	86.7	75.8	83.5
MSTN [137]	89.9	88.2	75.9	84.7
MCD [46]	88.1	86.3	73.4	82.7
SWD [92]	89.0	87.3	73.8	84.4
SSTDA	**90.0**	**89.1**	**78.0**	**86.2**
50Salads	F1@{10, 25, 50}			Edit
Source only (MS-TCN)	75.4	73.4	65.2	68.9
VCOP [102]	75.8	73.8	65.9	68.4
DANN [43]	79.2	77.8	70.3	72.0
JAN [41]	80.9	79.4	72.4	73.5
MADA [136]	79.6	77.4	70.0	72.4
MSTN [137]	79.3	77.6	71.5	72.1
MCD [46]	78.2	75.5	67.1	70.8
SWD [92]	78.2	76.2	67.4	71.6
SSTDA	**83.0**	**81.5**	**73.8**	**75.8**
Breakfast	F1@{10, 25, 50}			Edit
Source only (MS-TCN)	65.3	59.6	47.2	65.7
VCOP [102]	68.5	62.9	50.1	67.9
DANN [43]	72.8	67.8	55.1	71.7
JAN [41]	70.2	64.7	52.0	70.0
MADA [136]	71.0	65.4	52.8	71.2
MSTN [137]	69.6	63.6	51.5	69.2
MCD [46]	70.4	65.1	52.4	69.7
SWD [92]	68.6	63.2	50.6	69.1
SSTDA	**75.0**	**69.1**	**55.2**	**73.7**

comparable performance with MS-TCN, as shown in Table 5.4. We then evaluate the proposed SSTDA on GTEA and Breakfast with the same percentage of labeled training data, and also get comparable or better performance.

Comparison with the State of the Art: Finally, we compare the recent state of the art to SSTDA trained with the common fully-source-supervision settings.

This setting means we have labels for all the frames in source videos, and SSTDA

Table 5.4: The comparison of SSTDA trained with less labeled training data. m in the first row indicates the percentage of labeled training data used to train a model.

50Salads	m%	F1@{10, 25, 50}			Edit	Acc
SSTDA	100%	83.0	81.5	73.8	75.8	83.2
	95%	81.6	80.0	73.1	75.6	83.2
	85%	81.0	78.9	70.9	73.8	82.1
	75%	78.9	76.5	68.6	71.7	81.1
	65%	**77.7**	**75.0**	**66.2**	**69.3**	**80.7**
MS-TCN	100%	75.4	73.4	65.2	68.9	82.1
GTEA	m%	F1@{10, 25, 50}			Edit	Acc
SSTDA	100%	90.0	89.1	78.0	86.2	79.8
	65%	**85.2**	**82.6**	**69.3**	**79.6**	**75.7**
MS-TCN	100%	86.5	83.6	71.9	81.3	76.5
Breakfast	m%	F1@{10, 25, 50}			Edit	Acc
SSTDA	100%	75.0	69.1	55.2	73.7	70.2
	65%	**69.3**	**62.9**	**49.4**	**69.0**	**65.8**
MS-TCN	100%	65.3	59.6	47.2	65.7	64.7

outperforms all the previous methods on the three datasets with respect to all evaluation metrics. For example, SSTDA outperforms currently the state-of-the-art fully-supervised method, MS-TCN [33], by large margins e.g. 8.1% for F1@25, 8.6% for F1@50, and 6.9% for the edit score on 50Salads; 9.5% for F1@25, 8.0% for F1@50, and 8.0% for the edit score on Breakfast), as demonstrated in Table 5.5. Since no additional labeled data is used, these results indicate how our proposed SSTDA address the spatio-temporal variation problem with unlabeled videos to improve the action segmentation performance.

5.2.4 Ablation Study and Analysis

Design Choice for Local SSTDA: Since we develop our approaches upon MS-TCN [33], it raises the question: *How to effectively integrate binary domain prediction to a multi-stage architecture?* To answer this, we first integrate $\hat{G}_{ld}$ into each stage and the results show that the best performance happens when the $\hat{G}_{ld}$ is integrated into middle stages, such as $S2$ or $S3$, as shown in Table 5.6. $S1$ is not a good choice for DA because it corresponds to low-level features with less discriminability where DA shows limited effects [40], and

Table 5.5: Comparison with the state of the art on GTEA, 50Salads, and the Breakfast dataset.

GTEA	F1@{10, 25, 50}			Edit	Acc
ST-CNN [138]	58.7	54.4	41.9	49.1	60.6
Bi-LSTM [25]	66.5	59.0	43.6	-	55.5
ED-TCN [30]	72.2	69.3	56.0	-	64.0
LCDC [139]	75.4	-	-	72.8	65.3
TricorNet [31]	76.0	71.1	59.2	-	64.8
TDRN [32]	79.2	74.4	62.7	74.1	70.1
MS-TCN [33]†	86.5	83.6	71.9	81.3	76.5
SSTDA (65%)	85.2	82.6	69.3	79.6	75.7
SSTDA	**90.0**	**89.1**	**78.0**	**86.2**	**79.8**
50Salads	F1@{10, 25, 50}			Edit	Acc
ST-CNN [138]	55.9	49.6	37.1	45.9	59.4
ED-TCN [30]	68.0	63.9	52.6	59.8	64.7
TricorNet [31]	70.1	67.2	56.6	62.8	67.5
TDRN [32]	72.9	68.5	57.2	66.0	68.1
LCDC [139]	73.8	-	-	66.9	72.1
MS-TCN [33]†	75.4	73.4	65.2	68.9	82.1
SSTDA (65%)	77.7	75.0	66.2	69.3	80.7
SSTDA	**83.0**	**81.5**	**73.8**	**75.8**	**83.2**
Breakfast	F1@{10, 25, 50}			Edit	Acc
ED-TCN [30]	-	-	-	-	43.3
HTK [77]	-	-	-	-	50.7
TCFPN [79]	-	-	-	-	52.0
HTK (64) [140]	-	-	-	-	56.3
GRU [78]	-	-	-	-	60.6
MS-TCN [33]†	65.3	59.6	47.2	65.7	64.7
SSTDA (65%)	69.3	62.9	49.4	69.0	65.8
SSTDA	**75.0**	**69.1**	**55.2**	**73.7**	**70.2**

Table 5.6: The experimental results of design choice for local SSTDA (on GTEA).

	F1@{10, 25, 50}			Edit	Acc
Source only	86.5	83.6	71.9	81.3	76.5
$\{S1\}$	88.6	86.2	73.6	84.2	78.7
$\{S2\}$	89.1	87.2	**74.4**	84.3	79.1
$\{S3\}$	89.2	87.3	72.3	83.8	78.9
$\{S4\}$	88.1	86.4	73.0	83.0	78.8
$\{S1, S2\}$	89.0	85.8	73.5	**84.8**	79.5
$\{S2, S3\}$	**89.6**	**87.9**	**74.4**	84.5	**80.1**
$\{S3, S4\}$	88.3	86.8	73.9	83.6	78.6

Table 5.7: The experimental results for different segment numbers of sequential domain prediction (on GTEA).

Segment #	F1@$\{10, 25, 50\}$			Edit	Acc
1	89.4	87.7	75.4	85.3	79.2
2	**90.0**	**89.1**	**78.0**	**86.2**	**79.8**
3	89.7	87.6	75.4	85.2	79.2

represents less temporal receptive fields for videos. However, higher stages (e.g. $S4$) are not always better. We conjecture that it is because the model fits more to the source data, causing difficulty for DA. In our case, integrating $\hat{G}_{ld}$ into $S2$ provides the best overall performance.

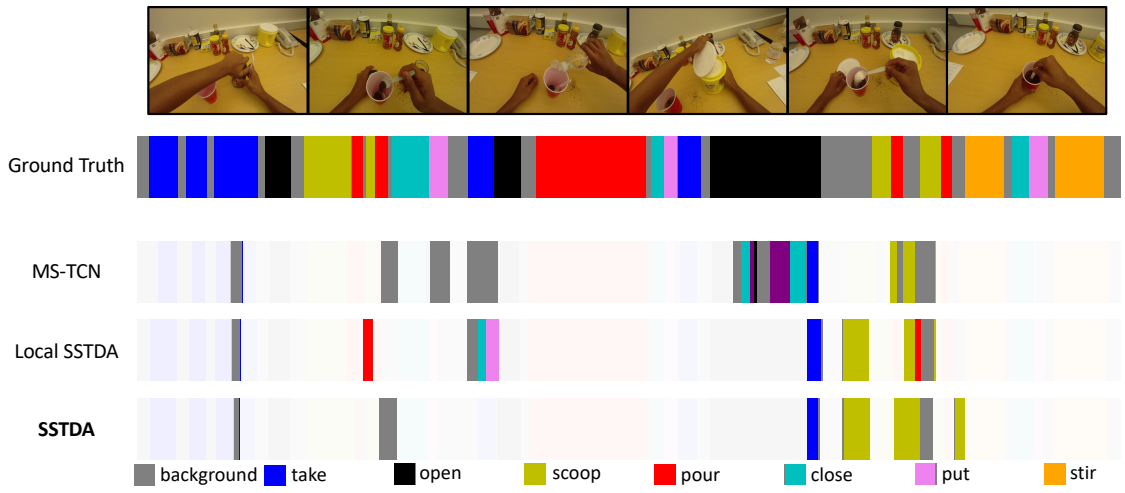
We also integrate binary domain prediction with multiple stages. However, multi-stage DA does not always guarantee improved performance. For example, $\{S1, S2\}$ has worse results than $\{S2\}$ in terms of F1@$\{10, 25, 50\}$. Since $\{S2\}$ and $\{S3\}$ provide the best single-stage DA performance, we use $\{S2, S3\}$, which performs the best, as the final model for all our approaches in all the experiments.

Design Choice for Global SSTDA: The most critical design decision for the sequential domain prediction is the segment number for each video. In our implementation, we divide one source video into m segments and do so for one target video, and then apply G_{gd} to predict the permutation of domains for these $2m$ video segments. Therefore, the category number of G_{gd} equals the number of all permutations $(2m)!/(m!)^2$. In other words, the segment number m determine the complexity of the self-supervised auxiliary task. For example, $m = 3$ leads to a 20-way classifier, and $m = 4$ results in a 70-way classifier. Since a good self-supervised task should be neither naive nor over complicated [96], we choose $m = 2$ as our final decision, which is supported by our experiments as shown in Table 5.7.

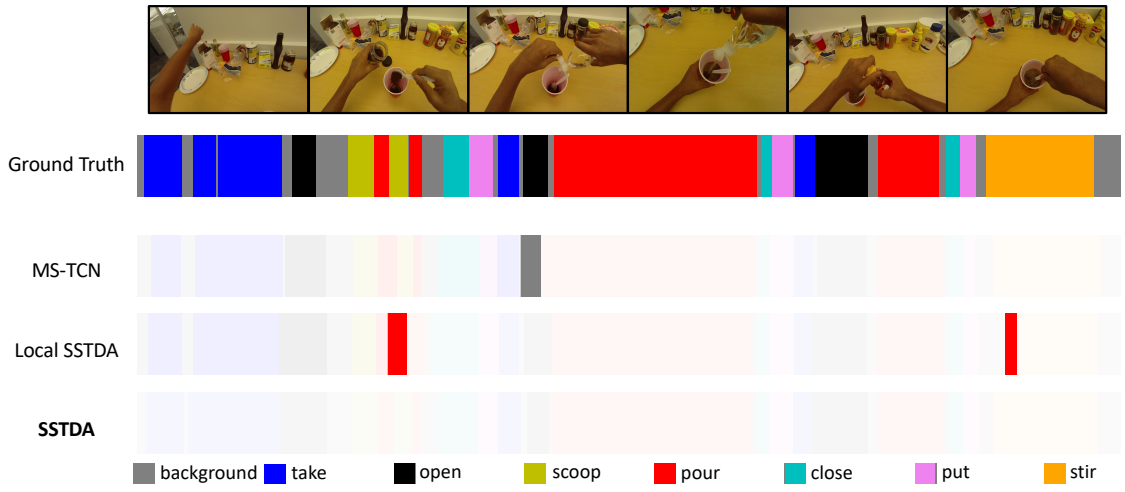
Segmentation Visualization: It is also common to evaluate the qualitative performance to ensure that the prediction results are aligned with human vision. First, we compare our approaches with the baseline model MS-TCN [33] and the ground truth, as shown in Fig-

ures 5.7 to 5.9. For example, local SSTDA falsely detects the *pour* action in Figure 5.7b, falsely classifies *cheese*-related actions as *cucumber*-related actions in Figure 5.8b, and falsely detects the *stir milk* action in Figure 5.9b. However, by jointly aligning local and global temporal dynamics with SSTDA, the model is effectively adapted to the target domain, reducing the above mentioned incorrect predictions and achieving better segmentation.

We then compare SSTDA with other DA methods, and Figure 5.10 shows that our result is the closest to the ground truth. The others either fail to detect some actions, falsely predict some actions that do not exist, or make incorrect classification.

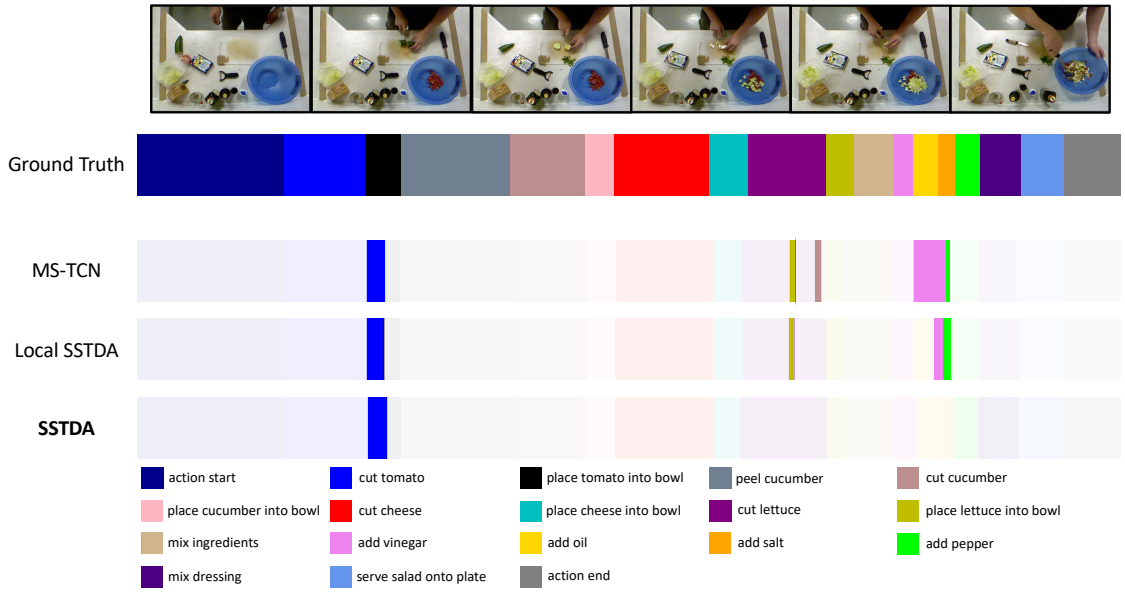


(a) *Make coffee*

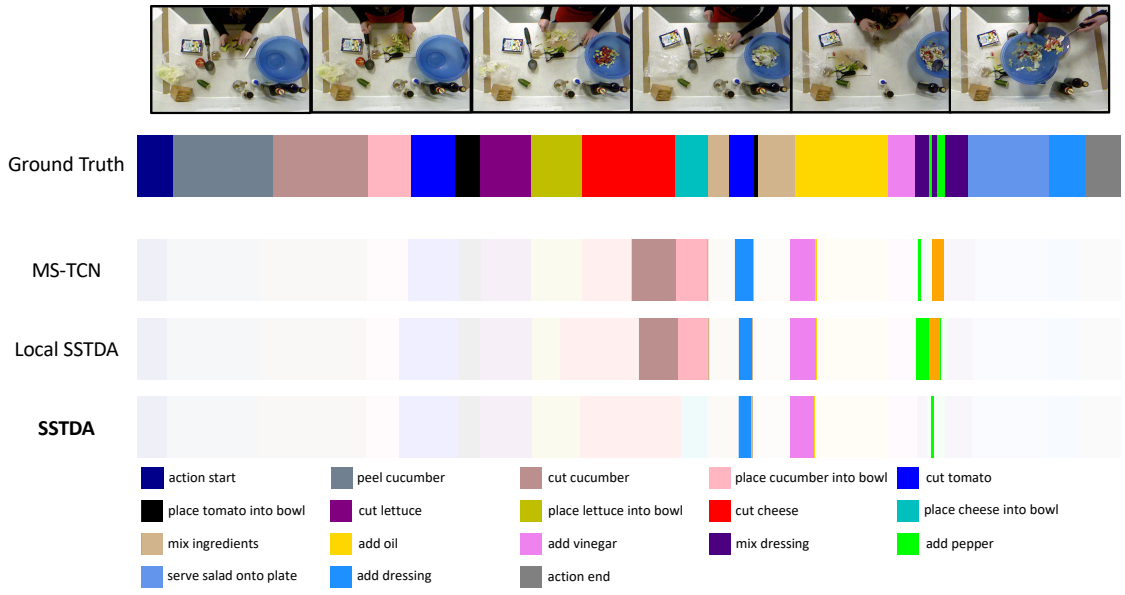


(b) *Make honey coffee*

Figure 5.7: The visualization of temporal action segmentation for our methods with color-coding on *GTEA*.

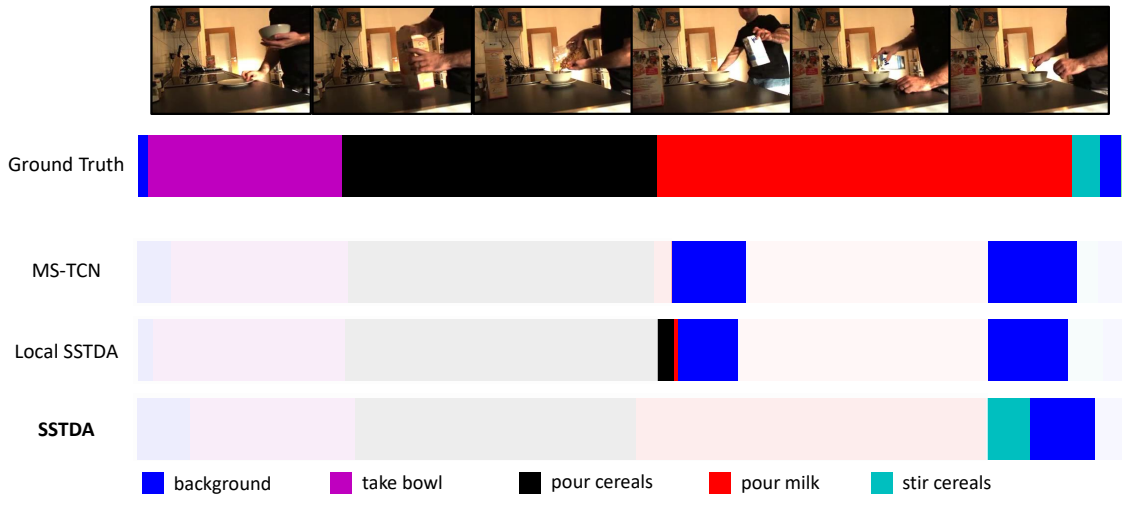


(a) *Subject 02*



(b) *Subject 04*

Figure 5.8: The visualization of temporal action segmentation for our methods with color-coding on *50Salads*.



(a) *Make Cereal*

(b) *Make milk*

Figure 5.9: The visualization of temporal action segmentation for our methods with color-coding on *Breakfast*.

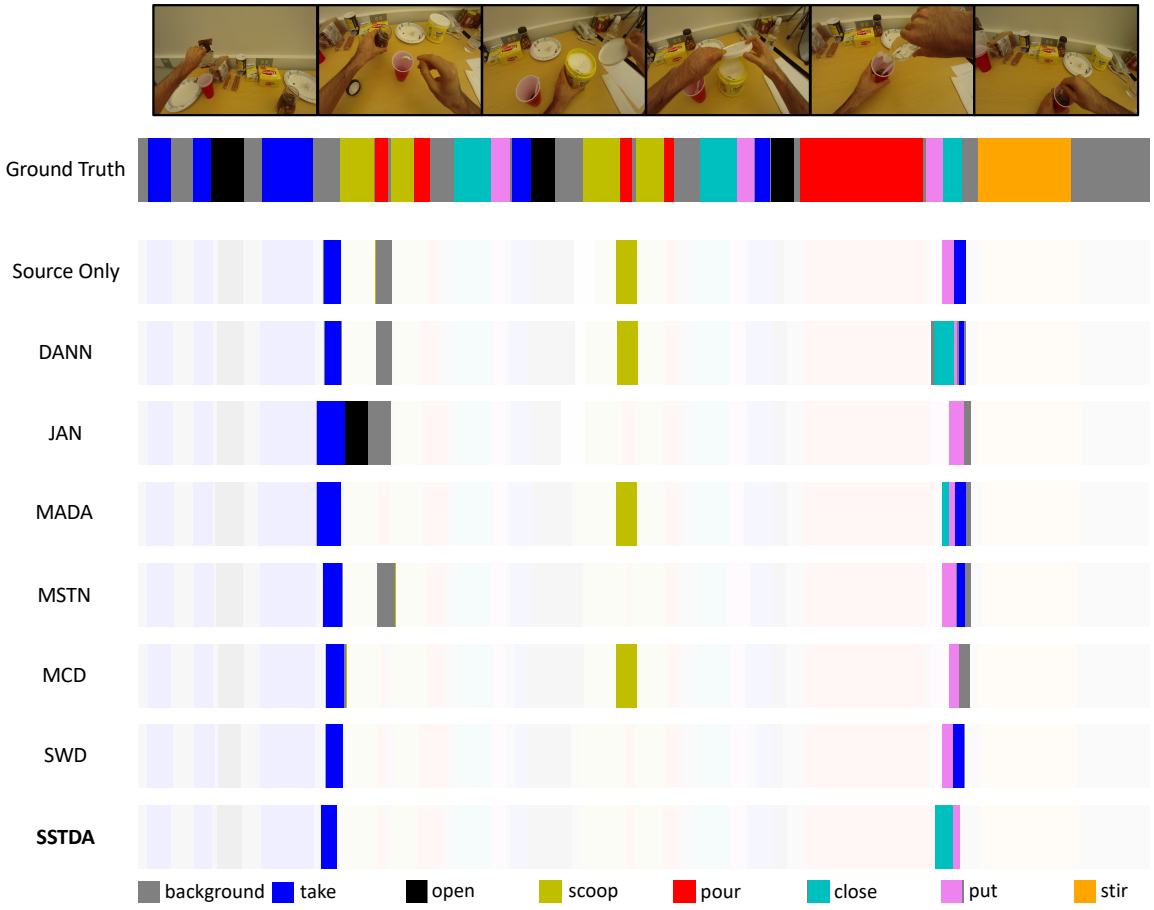


Figure 5.10: The visualization of temporal action segmentation for different DA methods. (input example: *Make coffee* from GTEA)

CHAPTER 6

CONCLUSIONS AND FUTURE DIRECTIONS

This dissertation investigates several methods to exploit temporal information for video understanding under full-supervised and cross-domain settings. We start by comprehensive background information about video understanding and cross-domain learning. We then explore the importance of temporal information and develop novel approaches that integrating temporal information to tackle various video tasks. The contributions of this thesis are three-fold. Firstly, we focus on the general video classification task, and thoroughly explore two methods to model dynamic temporal information: **Temporal Segment LSTM** and **Temporal-Inception**. We show that naive temporal max-pooling performed similar to the vanilla LSTM. By integrating temporal segments and LSTM, the proposed method can exploit temporal information more effectively. On the other hand, our proposed Temporal-Inception performs convolutions on temporally-constructed feature vectors to learn global video-level representations. Both approaches achieve comparable state-of-the-art accuracy on both the UCF101 and HMDB51 datasets using only high-level feature vector representations equally sampled from each of the videos, demonstrating that both RNNs and convolutions across time are able to model temporal dynamics. With systematic analyses, we identify the key components for each method that contribute to performance improvement, facilitating research in this field. Secondly, we investigate how temporal information can help address domain shift problem in the cross-domain action classification task. we propose two large-scale cross-domain datasets for action classification, **UCF-HMDB**$_{full}$ and **Kinetics-Gameplay**, including both real and virtual domains. We use these datasets to investigate the domain shift problem across videos, and show that simultaneously aligning and learning temporal dynamics achieves effective alignment without the need for sophisticated DA methods. By building upon the above finding, we propose **Temporal Attentive**

Adversarial Adaptation Network (TA^{3}N) to simultaneously attend, align and learn temporal dynamics across domains, achieving state-of-the-art performance on all of the cross-domain video datasets investigated. Finally, we investigate how temporal information can address the problem of spatio-temporal intra-class variations in a more challenging task, action segmentation. To effectively exploit the temporal information in unlabeled videos without labels, we propose **Self-Supervised Temporal Domain Adaptation (SSTDA)** to jointly align cross-domain feature spaces embedded with local and global temporal dynamics by two self-supervised auxiliary tasks, *binary* and *sequential domain prediction*. Our experiments indicate that SSTDA outperforms other DA approaches by aligning temporal dynamics more effectively. We also validate the proposed SSTDA on three challenging datasets (GTEA, 50Salads, and Breakfast), and show that SSTDA outperforms the current state-of-the-art method by large margins and only requires 65% of the labeled training data to achieve the comparable performance, demonstrating the usefulness of adapting to unlabeled videos across variations.

6.1 Future Research Directions

Video understanding has been researched for years, including a variety of tasks. We believe that temporal information is the key factor that we can exploit to benefit various tasks. This work will continue investigating different video tasks, such as spatio-temporal action localization [141], which requires locating action regions along time, and video object segmentation [142], which requires to segment objects along time in videos.

In this work, we mainly investigate the RGB data of videos, which is one of the information that humans utilize to understand videos. Therefore, we will investigate temporal information under different types of data (i.e. modalities), such as sound, text, etc., facilitating the understanding of how humans integrate different modalities to understand videos.

Finally, since the ultimate goal of our research is to solve real-world problems, we

would like to extend this work to different experimental settings which are closer to real-world scenarios, such as the open-set setting [143, 144, 36, 145], and multi-source domain adaptation [146, 147], etc.

REFERENCES

[1] https://www.youtube.com/yt/about/press/.

[2] https://www.alexa.com/siteinfo/youtube.com.

[3] G. A. Sigurdsson, O. Russakovsky, and A. Gupta, "What actions are needed for understanding human actions in videos?" In *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2137–2146.

[4] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," in *IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[5] C.-Y. Ma, A. Kadav, I. Melvin, Z. Kira, G. AlRegib, and H. P. Graf, "Attend and interact: Higher-order object interactions for video understanding," in *IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[6] C.-Y. Ma, M.-H. Chen, Z. Kira, and G. AlRegib, "Ts-lstm and temporal-inception: Exploiting spatiotemporal dynamics for activity recognition," *Signal Processing: Image Communication*, vol. 71, pp. 76–87, 2019.

[7] Y. Kong and Y. Fu, "Human action recognition and prediction: A survey," *arXiv preprint arXiv:1806.11230*, 2018.

[8] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, *et al.*, "The kinetics human action video dataset," *arXiv preprint arXiv:1705.06950*, 2017.

[9] J. Carreira, E. Noland, A. Banki-Horvath, C. Hillier, and A. Zisserman, "A short note about kinetics-600," *arXiv preprint arXiv:1808.01340*, 2018.

[10] J. Carreira, E. Noland, C. Hillier, and A. Zisserman, "A short note on the kinetics-700 human action dataset," *arXiv preprint arXiv:1907.06987*, 2019.

[11] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[12] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2014.

[13] 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 *IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[14] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *IEEE International Conference on Computer Vision (ICCV)*, 2015.

[15] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, "Beyond short snippets: Deep networks for video classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 4694–4702.

[16] C. Feichtenhofer, A. Pinz, and A. Zisserman, "Convolutional two-stream network fusion for video action recognition," in *IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[17] 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, pp. 20–36.

[18] J. Carreira and A. Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset," in *IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[19] Z. Qiu, T. Yao, and T. Mei, "Learning spatio-temporal representation with pseudo-3d residual networks," in *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[20] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri, "A closer look at spatiotemporal convolutions for action recognition," in *IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[21] B. Zhou, A. Andonian, A. Oliva, and A. Torralba, "Temporal relational reasoning in videos," in *European Conference on Computer Vision (ECCV)*, 2018.

[22] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre, "Hmdb: A large video database for human motion recognition," in *IEEE International Conference on Computer Vision (ICCV)*, 2011.

[23] K. Soomro, A. R. Zamir, and M. Shah, "Ucf101: A dataset of 101 human actions classes from videos in the wild," *arXiv preprint arXiv:1212.0402*, 2012.

[24] S. Abu-El-Haija, N. Kothari, J. Lee, P. Natsev, G. Toderici, B. Varadarajan, and S. Vijayanarasimhan, "Youtube-8m: A large-scale video classification benchmark," *arXiv preprint arXiv:1609.08675*, 2016.

[25] B. Singh, T. K. Marks, M. Jones, O. Tuzel, and M. Shao, "A multi-stream bi-directional recurrent neural network for fine-grained action detection," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[26] B. Ni, X. Yang, and S. Gao, "Progressively parsing interactional objects for fine grained action detection," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[27] D.-A. Huang, L. Fei-Fei, and J. C. Niebles, "Connectionist temporal modeling for weakly supervised action labeling," in *European Conference on Computer Vision (ECCV)*, 2016.

[28] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *International Conference on Machine Learning (ICML)*, 2013.

[29] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[30] C. Lea, M. D. Flynn, R. Vidal, A. Reiter, and G. D. Hager, "Temporal convolutional networks for action segmentation and detection," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[31] L. Ding and C. Xu, "Tricornet: A hybrid temporal convolutional and recurrent network for video action segmentation," *arXiv preprint arXiv:1705.07818*, 2017.

[32] P. Lei and S. Todorovic, "Temporal deformable residual networks for action segmentation in videos," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[33] Y. A. Farha and J. Gall, "Ms-tcn: Multi-stage temporal convolutional network for action segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[34] D.-A. Huang, V. Ramanathan, D. Mahajan, L. Torresani, M. Paluri, L. Fei-Fei, and J. Carlos Niebles, "What makes a video a video: Analyzing temporal information in video understanding models and datasets," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7366–7375.

[35] J. Quionero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence, *Dataset Shift in Machine Learning*. The MIT Press, 2009.

[36] X. Peng, B. Usman, K. Saito, N. Kaushik, J. Hoffman, and K. Saenko, "Syn2real: A new benchmark for synthetic-to-real visual domain adaptation," *arXiv preprint arXiv:1806.09755*, 2018.

[37] S. J. Pan, Q. Yang, *et al.*, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, vol. 22, no. 10, pp. 1345–1359, 2010.

[38] G. Csurka, "A comprehensive survey on domain adaptation for visual applications," in *Domain Adaptation in Computer Vision Applications*, Springer, 2017, pp. 1–35.

[39] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, "Decaf: A deep convolutional activation feature for generic visual recognition," in *International Conference on Machine Learning (ICML)*, 2014.

[40] M. Long, Y. Cao, J. Wang, and M. Jordan, "Learning transferable features with deep adaptation networks," in *International Conference on Machine Learning (ICML)*, 2015.

[41] M. Long, H. Zhu, J. Wang, and M. I. Jordan, "Deep transfer learning with joint adaptation networks," in *International Conference on Machine Learning (ICML)*, 2017.

[42] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," in *International Conference on Machine Learning (ICML)*, 2015.

[43] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 2096–2030, 2016.

[44] Y. Li, N. Wang, J. Shi, J. Liu, and X. Hou, "Revisiting batch normalization for practical domain adaptation," in *International Conference on Learning Representations Workshop (ICLRW)*, 2017.

[45] Y. Li, N. Wang, J. Shi, X. Hou, and J. Liu, "Adaptive batch normalization for practical domain adaptation," *Pattern Recognition*, vol. 80, pp. 109–117, 2018.

[46] K. Saito, K. Watanabe, Y. Ushiku, and T. Harada, "Maximum classifier discrepancy for unsupervised domain adaptation," in *IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[47] K. Saenko, B. Kulis, M. Fritz, and T. Darrell, "Adapting visual category models to new domains," in *European Conference on Computer Vision (ECCV)*, Springer, 2010, pp. 213–226.

[48] W. Sultani and I. Saleemi, "Human action recognition across datasets by foreground-weighted histogram decomposition," in *IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[49] T. Xu, F. Zhu, E. K. Wong, and Y. Fang, "Dual many-to-one-encoder-based transfer learning for cross-dataset human action recognition," *Image and Vision Computing*, vol. 55, pp. 127–137, 2016.

[50] A. Jamal, V. P. Namboodiri, D. Deodhare, and K. Venkatesh, "Deep domain adaptation in action space," in *British Machine Vision Conference (BMVC)*, 2018.

[51] X.-Y. Zhang, H. Shi, C. Li, K. Zheng, X. Zhu, and L. Duan, "Learning transferable self-attentive representations for action recognition in untrimmed videos with weak supervision," in *AAAI Conference on Artificial Intelligence (AAAI)*, 2019.

[52] A. Lahiri, S. C. Ragireddy, P. Biswas, and P. Mitra, "Unsupervised adversarial visual level domain adaptation for learning video object detectors from images," in *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2019.

[53] A. Fathi, X. Ren, and J. M. Rehg, "Learning to recognize objects in egocentric activities," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[54] S. Stein and S. J. McKenna, "Combining embedded accelerometers with computer vision for recognizing food preparation activities," in *ACM international joint conference on Pervasive and ubiquitous computing (UbiComp)*, 2013.

[55] H. Kuehne, A. Arslan, and T. Serre, "The language of actions: Recovering the syntax and semantics of goal-directed human activities," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[56] H. Wang, M. M. Ullah, A. Klaser, I. Laptev, and C. Schmid, "Evaluation of local spatio-temporal features for action recognition," *Proceedings of the British Machine Vision Conference (BMVC)*, 2009.

[57] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu, "Action recognition by dense trajectories," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 2011, pp. 3169–3176.

[58] H. Wang and C. Schmid, "Action recognition with improved trajectories," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 3551–3558.

[59] H. Li, J. Chen, Z. Xu, H. Chen, and R. Hu, "Multiple instance discriminative dictionary learning for action recognition," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 2014–2018.

[60] Q. Huang, S. Sun, and F. Wang, "A compact pairwise trajectory representation for action recognition," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 1767–1771.

[61] X. Peng, L. Wang, X. Wang, and Y. Qiao, "Bag of visual words and fusion methods for action recognition: Comprehensive study and good practice," *Computer Vision and Image Understanding*, 2016.

[62] K. Xu, X. Jiang, and T. Sun, "Two-stream dictionary learning architecture for action recognition," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 3, pp. 567–576, 2017.

[63] D. Roy, K. S. R. Murty, and C. K. Mohan, "Action-vectors: Unsupervised movement modeling for action recognition," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 1602–1606.

[64] L. Zhang, Y. Feng, X. Xiang, and X. Zhen, "Realistic human action recognition: When cnns meet lds," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 1622–1626.

[65] B. Banerjee and V. Murino, "Efficient pooling of image based cnn features for action recognition in videos," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 2637–2641.

[66] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR*, 2015.

[67] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems (NeurIPS)*, 2012.

[68] L. Sun, K. Jia, D.-Y. Yeung, and B. E. Shi, "Human action recognition using factorized spatio-temporal convolutional networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 4597–4605.

[69] S. Yan, J. S. Smith, W. Lu, and B. Zhang, "Hierarchical multi-scale attention networks for action recognition," *Signal Processing: Image Communication*, vol. 61, pp. 73–84, 2018.

[70] P. Pan, Z. Xu, Y. Yang, F. Wu, and Y. Zhuang, "Hierarchical recurrent neural encoder for video representation with application to captioning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1029–1038.

[71] P. Wang, Y. Cao, C. Shen, L. Liu, and H. T. Shen, "Temporal pyramid pooling-based convolutional neural network for action recognition," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 12, pp. 2613–2622, 2017.

[72] Y. Wang, J. Song, L. Wang, L. Van Gool, and O. Hilliges, "Two-stream sr-cnns for action recognition in videos," *Proceedings of the British Machine Vision Conference (BMVC)*, 2016.

[73] W. Zhu, J. Hu, G. Sun, X. Cao, and Y. Qiao, "A key volume mining deep framework for action recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1991–1999.

[74] X. Wang, A. Farhadi, and A. Gupta, "Actions transformations," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2658–2667.

[75] S. Zhao, Y. Liu, Y. Han, R. Hong, Q. Hu, and Q. Tian, "Pooling the convolutional layers in deep convnets for video action recognition," *IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)*, vol. 28, no. 8, pp. 1839–1849, 2017.

[76] H. Jégou, M. Douze, C. Schmid, and P. Pérez, "Aggregating local descriptors into a compact image representation," in *IEEE computer society conference on computer vision and pattern recognition (CVPR)*, 2010.

[77] H. Kuehne, A. Richard, and J. Gall, "Weakly supervised learning of actions from transcripts," *Computer Vision and Image Understanding (CVIU)*, vol. 163, pp. 78–89, 2017.

[78] A. Richard, H. Kuehne, and J. Gall, "Weakly supervised action learning with rnn based fine-to-coarse modeling," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[79] L. Ding and C. Xu, "Weakly-supervised action segmentation with iterative soft boundary assignment," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[80] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell, "Deep domain confusion: Maximizing for domain invariance," *arXiv preprint arXiv:1412.3474*, 2014.

[81] M. Long, H. Zhu, J. Wang, and M. I. Jordan, "Unsupervised domain adaptation with residual transfer networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

[82] W. Zellinger, T. Grubinger, E. Lughofer, T. Natschläger, and S. Saminger-Platz, "Central moment discrepancy (cmd) for domain-invariant representation learning," in *International Conference on Learning Representations (ICLR)*, 2017.

[83] H. Yan, Y. Ding, P. Li, Q. Wang, Y. Xu, and W. Zuo, "Mind the class weight bias: Weighted maximum mean discrepancy for unsupervised domain adaptation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[84] B. Sun and K. Saenko, "Deep coral: Correlation alignment for deep domain adaptation," in *European Conference on Computer Vision Workshop (ECCVW)*, 2016.

[85] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2014.

[86] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, "Adversarial discriminative domain adaptation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[87] W. Zhang, W. Ouyang, W. Li, and D. Xu, "Collaborative and adversarial network for unsupervised domain adaptation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

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

[89] F. M. Carlucci, L. Porzi, B. Caputo, E. Ricci, and S. R. Bulò, "Autodial: Automatic domain alignment layers," in *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[90] G. French, M. Mackiewicz, and M. Fisher, "Self-ensembling for visual domain adaptation," in *International Conference on Learning Representations (ICLR)*, 2018.

[91] K. Saito, Y. Ushiku, T. Harada, and K. Saenko, "Adversarial dropout regularization," in *International Conference on Learning Representations (ICLR)*, 2018.

[92] C.-Y. Lee, T. Batra, M. H. Baig, and D. Ulbricht, "Sliced wasserstein discrepancy for unsupervised domain adaptation," in *IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[93] X. Wang, L. Li, W. Ye, M. Long, and J. Wang, "Transferable attention for domain adaptation," in *AAAI Conference on Artificial Intelligence (AAAI)*, 2019.

[94] V. K. Kurmi, S. Kumar, and V. P. Namboodiri, "Attending to discriminative certainty for domain adaptation," in *IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[95] C. Doersch, A. Gupta, and A. A. Efros, "Unsupervised visual representation learning by context prediction," in *IEEE International Conference on Computer Vision (ICCV)*, 2015.

[96] M. Noroozi and P. Favaro, "Unsupervised learning of visual representations by solving jigsaw puzzles," in *European Conference on Computer Vision (ECCV)*, 2016.

[97] S. Gidaris, P. Singh, and N. Komodakis, "Unsupervised representation learning by predicting image rotations," in *International Conference on Learning Representations (ICLR)*, 2018.

[98] H.-Y. Lee, J.-B. Huang, M. Singh, and M.-H. Yang, "Unsupervised representation learning by sorting sequences," in *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[99] D. Wei, J. J. Lim, A. Zisserman, and W. T. Freeman, "Learning and using the arrow of time," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[100] D. Kim, D. Cho, and I. S. Kweon, "Self-supervised video representation learning with space-time cubic puzzles," in *AAAI Conference on Artificial Intelligence (AAAI)*, 2019.

[101] U. Ahsan, R. Madhok, and I. Essa, "Video jigsaw: Unsupervised learning of spatiotemporal context for video action recognition," in *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2019.

[102] D. Xu, J. Xiao, Z. Zhao, J. Shao, D. Xie, and Y. Zhuang, "Self-supervised spatiotemporal learning via video clip order prediction," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[103] R. Collobert, K. Kavukcuoglu, and C. Farabet, "Torch7: A matlab-like environment for machine learning," *BigLearn, NeurIPS Workshop*, 2011.

[104] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[105] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[106] Y. Pan, T. Mei, T. Yao, H. Li, and Y. Rui, "Jointly modeling embedding and translation to bridge video and language," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4594–4602.

[107] L. Wang, Y. Xiong, Z. Wang, and Y. Qiao, "Towards good practices for very deep two-stream convnets," *arXiv preprint arXiv:1507.02159*, 2015.

[108] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert, "High accuracy optical flow estimation based on a theory for warping," in *European conference on computer vision*, Springer, 2004, pp. 25–36.

[109] C. Zach, T. Pock, and H. Bischof, "A duality based approach for realtime tv-l 1 optical flow," in *Joint Pattern Recognition Symposium*, Springer, 2007, pp. 214–223.

[110] C. Feichtenhofer, A. Pinz, and R. P. Wildes, "Spatiotemporal multiplier networks for video action recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 2017, pp. 7445–7454.

[111] Y. Wang, M. Long, J. Wang, and P. S. Yu, "Spatiotemporal pyramid network for video action recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 2017, pp. 2097–2106.

[112] A. Diba, V. Sharma, and L. Van Gool, "Deep temporal linear encoding networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1541–1550.

[113] N. Srivastava, E. Mansimov, and R. Salakhudinov, "Unsupervised learning of video representations using lstms," in *International Conference on Machine Learning*, 2015, pp. 843–852.

[114] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 4489–4497.

[115] X. Wang, C. Qi, and F. Lin, "Combined trajectories for action recognition based on saliency detection and motion boundary," *Signal Processing: Image Communication*, vol. 57, pp. 91–102, 2017.

[116] L. Wang, Y. Qiao, and X. Tang, "Action recognition with trajectory-pooled deep-convolutional descriptors," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 4305–4314.

[117] Z. Li, W. Wang, N. Li, and J. Wang, "Tube convnets: Better exploiting motion for action recognition," in *Image Processing (ICIP), 2016 IEEE International Conference on*, IEEE, 2016, pp. 3056–3060.

[118] Q.-Q. Chen, F. Liu, X. Li, B.-D. Liu, and Y.-J. Zhang, "Saliency-context two-stream convnets for action recognition," in *Image Processing (ICIP), 2016 IEEE International Conference on*, IEEE, 2016, pp. 3076–3080.

[119] Z. Wu, Y.-G. Jiang, X. Wang, H. Ye, and X. Xue, "Multi-stream multi-class fusion of deep networks for video classification," in *Proceedings of the 2016 ACM on Multimedia Conference*, ACM, 2016, pp. 791–800.

[120] C. Feichtenhofer, A. Pinz, and R. Wildes, "Spatiotemporal residual networks for video action recognition," in *Advances in Neural Information Processing Systems*, 2016, pp. 3468–3476.

[121] N. Passalis and A. Tefas, "Learning bag-of-features pooling for deep convolutional neural networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 5755–5763.

[122] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "Netvlad: Cnn architecture for weakly supervised place recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5297–5307.

[123] R. Girdhar, D. Ramanan, A. Gupta, J. Sivic, and B. Russell, "Actionvlad: Learning spatio-temporal aggregation for action classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, 2017, p. 3.

[124] X. Long, C. Gan, G. de Melo, J. Wu, X. Liu, and S. Wen, "Attention clusters: Purely attention based local feature integration for video classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7834–7843.

[125] M. Dehghani, S. Gouws, O. Vinyals, J. Uszkoreit, and L. Kaiser, "Universal transformers," *arXiv preprint arXiv:1807.03819*, 2018.

[126] 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," in *Advances in Neural Information Processing Systems Workshop (NeurIPSW)*, 2017.

[127] B. Steiner, Z. DeVito, S. Chintala, S. Gross, A. Paszke, F. Massa, A. Lerer, G. Chanan, Z. Lin, E. Yang, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[128] M.-H. Chen, Z. Kira, G. AlRegib, J. Yoo, R. Chen, and J. Zheng, "Temporal attentive alignment for large-scale video domain adaptation," in *IEEE International Conference on Computer Vision (ICCV)*, 2019.

[129] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[130] A. Santoro, D. Raposo, D. G. Barrett, M. Malinowski, R. Pascanu, P. Battaglia, and T. Lillicrap, "A simple neural network module for relational reasoning," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[131] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *The Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[132] X. Wang, A. Jabri, and A. A. Efros, "Learning correspondence from the cycle-consistency of time," in *IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[133] D. Dwibedi, Y. Aytar, J. Tompson, P. Sermanet, and A. Zisserman, "Temporal cycle-consistency learning," in *IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[134] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[135] M.-H. Chen, B. Li, Y. Bao, G. AlRegib, and Z. Kira, "Action segmentation with joint self-supervised temporal domain adaptation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[136] Z. Pei, Z. Cao, M. Long, and J. Wang, "Multi-adversarial domain adaptation," in *AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

[137] S. Xie, Z. Zheng, L. Chen, and C. Chen, "Learning semantic representations for unsupervised domain adaptation," in *International Conference on Machine Learning (ICML)*, 2018.

[138] C. Lea, A. Reiter, R. Vidal, and G. D. Hager, "Segmental spatiotemporal cnns for fine-grained action segmentation," in *European Conference on Computer Vision (ECCV)*, 2016.

[139] K.-N. C. Mac, D. Joshi, R. A. Yeh, J. Xiong, R. S. Feris, and M. N. Do, "Learning motion in feature space: Locally-consistent deformable convolution networks for fine-grained action detection," in *IEEE International Conference on Computer Vision (ICCV)*, 2019.

[140] H. Kuehne, J. Gall, and T. Serre, "An end-to-end generative framework for video segmentation and recognition," in *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2016.

[141] C. Gu, C. Sun, D. A. Ross, C. Vondrick, C. Pantofaru, Y. Li, S. Vijayanarasimhan, G. Toderici, S. Ricco, R. Sukthankar, *et al.*, "Ava: A video dataset of spatio-temporally localized atomic visual actions," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[142] S. Caelles, J. Pont-Tuset, F. Perazzi, A. Montes, K.-K. Maninis, and L. Van Gool, "The 2019 davis challenge on vos: Unsupervised multi-object segmentation," *arXiv preprint arXiv:1905.00737*, 2019.

[143] P. P. Busto and J. Gall, "Open set domain adaptation.," in *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[144] K. Saito, S. Yamamoto, Y. Ushiku, and T. Harada, "Open set domain adaptation by backpropagation," in *European Conference on Computer Vision (ECCV)*, 2018.

[145] Y.-C. Hsu, Z. Lv, and Z. Kira, "Learning to cluster in order to transfer across domains and tasks," in *International Conference on Learning Representations (ICLR)*, 2018.

[146] H. Zhao, S. Zhang, G. Wu, J. M. Moura, J. P. Costeira, and G. J. Gordon, "Adversarial multiple source domain adaptation," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[147] X. Peng, Q. Bai, X. Xia, Z. Huang, K. Saenko, and B. Wang, "Moment matching for multi-source domain adaptation," in *IEEE International Conference on Computer Vision (ICCV)*, 2019.

VITA

Min-Hung Chen obtained his Ph.D. from the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA in 2020. He worked under the supervision of Prof. Ghassan AlRegib and conducted research on topics related to Computer Vision and Machine Learning. His doctoral work is focused on *Video Understanding beyond Fully Supervision*. Mainly, his doctoral research and thesis are investigating and exploiting temporal dynamics from videos to tackle distributional discrepancy problems for video understanding, generalizing spatio-temporal models to different environments without fully-supervision.

He obtained his B.Sc. and M.Sc. in 2010 and 2012, respectively. His master's research was focused on developing noise-robust depth estimation algorithms for pinhole-masked light field cameras. In 2017, he worked at Aipoly as a deep learning engineer intern. He also worked as a research scientist intern at Sony Interactive Entertainment in 2018 and at Baidu USA in 2019, respectively. His research interests span transfer learning, domain adaptation, machine learning beyond fully-supervision, video understanding, computer vision, deep learning, image and video processing, and all the related applications.

He published several papers at top conferences in the field of Computer Vision and Machine Learning, such as *CVPR, ICCV, WACV*, including poster and oral papers. He also served as reviewers for various conferences and journals, such as *ECCV, NeurIPS, T-CSVT*. He is the recipient of the ICCV 2019 Student Travel Grant Award. He also received Ministry of Education Technologies Incubation Scholarship from Taiwan from 2014 to 2017, and Otto F. and Jenny H. Krauss Fellowship from Georgia Institute of Technology during 2014 and 2015.