ulm university universität ulm

Human Action Parsing in Untrimmed Videos and Its Applications for Elderly People Healthcare

Dissertation
zur Erlangung des Doktorgrades **Dr.rer.nat.**
der Fakultät für Ingenieurwissenschaften, Informatik und Psychologie der Universität Ulm

Vorgelegt von
Yan Zhang
aus Qingzhou, Shandong, China
Institut für Neuroinformatik
Ulm, 2020

Amtierender Dekan
Prof. Dr. Maurits Ortmanns

Gutachter
Prof. Dr. Heiko Neumann
Prof. Dr. Siyu Tang

Tag der Promotion
Juni 24, 2020

Parts of this dissertation have already been made available for the scientific public:

Zhang, Y., Sun, H., Tang, S., Neumann, H. (2018). Temporal human action segmentation via dynamic clustering. arXiv preprint arXiv:1803.05790.

Zhang, Y., Tang, S., Sun, H., Neumann, H. (2018, September). Human Motion Parsing by Hierarchical Dynamic Clustering. In BMVC (p. 269).

Zhang Y., Neumann H. (2019) An Empirical Study Towards Understanding How Deep Convolutional Nets Recognize Falls. In: Leal-Taixé L., Roth S. (eds) Computer Vision – ECCV 2018 Workshops. ECCV 2018. Lecture Notes in Computer Science, vol 11134. Springer, Cham. Doi:10.1007/978-3-030-11024-6_8.

Zhang, Y., Tang, S., Muandet, K., Jarvers, C., Neumann, H. (2019). Local temporal bilinear pooling for fine-grained action parsing. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 12005-12015). DOI: 10.1109/CVPR.2019.01228

Zhang, Y., Muandet, K., Ma, Q., Neumann, H., Tang, S. (2019). Frontal Low-rank Random Tensors for Fine-grained Action Segmentation. arXiv preprint arXiv:1906.01004.

Abstract

Motivated by the demands and desires to improve the living quality of elderly people, in this thesis, we investigate visual human behavior analysis, and aim at proposing effective and reliable healthcare solutions for elderly people in various scenarios. Specifically, we focus on human behavior understanding from videos, captured by cameras in indoor environments, individual persons and actions at multiple granularity levels. When human behaviors can be understood automatically and reliably by computers, the living environment will become smart, provide effective assistance, and make the interaction between elderly people and smart environments as convenient as the interaction between youngsters and conventional environments.

Human behavior understanding from videos covers a broad range of tasks. To unify all the tasks, we propose the action parsing task: Given an untrimmed video with various types of actions, action parsing is to assign each individual frame an action label/cluster ID. Consequently, several classical tasks, e.g. action detection, video retrieval, action recognition and temporal action segmentation are unified within one framework. In this thesis, we first propose an unsupervised method, assigning each frame in the input video an cluster ID, and then a deep learning-based supervised method, assigning each frame an action label in an end-to-end manner. The unsupervised method is hierarchical dynamic clustering, which incorporates several novel modules and is inspired by the conventional *bag-of-visual-words* method for action recognition. For the deep learning-based supervised action parsing method, we employ a spatiotemporal convolutional encoder-decoder network, and propose novel bilinear pooling methods to realize fine-grained action parsing. We have applied the proposed methods to unsupervised action segmentation, abnormality (fainting) detection from omni-directional videos, explanation of how a deep neural network understands falls from videos, fine-grained human action parsing in daily living scenarios (recordings from both the third-part view and the egocentric view), and behavior understanding for surgical robots. All experimental results show that our proposed methods are effective and yield state-of-the-art performances.

Acknowledgment

Firstly, I sincerely express my gratitude and respect to my supervisor Prof. Dr. Heiko Neumann for the continuous support of my Ph.D study and related researches, for his patience, motivation, and immense knowledge. His guidance and supports helped me in all the time of research and writing of this thesis. Besides my supervisor, my sincere thanks go to the dissertation commission member: Prof. Dr. Siyu Tang, Dr. Friedhelm Schwenker, Prof. Dr. Wolfgang Minker, and Prof. Dr. Manfred Reichert. I faithfully appreciate their participation, and insightful comments on my work.

Also, sincerely thank Dr. Friedhelm Schwenker for offering me insightful ideas and numerous kind supports on affective computing researches. Sincerely thank Prof. Dr. Günther Palm, for providing knowledge and deep insights about artificial neural networks. Sincerely thank Dr. Steffen Walter, for offering many fruitful discussions and research facilities on studies of healthcare. Sincerely thank Prof. Dr. Enrico Rukzio, for providing insightful comments on my dissertation content.

In addition, my sincere thanks go to Dr. He Sun, who offered me numerous supports on developing dynamic clustering algorithms, writing papers, sharing ideas and making great discussions. My sincere thanks go to Prof. Dr. Siyu Tang and Prof. Dr. Michael J. Black, for providing me an opportunity to join their team, giving access to the laboratory and research facilities, and offering many inspiring discussions. My sincere thanks go to Dr. Krikamol Muandet, for providing insightful ideas and many kind supports to develop the temporal bilinear pooling theories.

I thank my fellow labmates in the institute of neural information processing (Ulm university), and labmates in Max Planck institute for intelligent systems (Tübingen) for the stimulating discussions, for the sleepless nights we were working together before deadlines, and for all the fun we have had in the past. Also I thank my friends and previous labmates in DKFZ Heidelberg, Saarland university and Max-planck institute for informatics (Saarbrücken).

Last but not the least, I would like to thank my parents and family for supporting me spiritually throughout writing this thesis and my life in general.

Contents

1Introduction

Improving the life quality of elderly people is one of the most important tasks in the world. As the development of medical conditions and living qualities, most countries all over the world have rising life expectations and population aging. As predicted in several reports, e.g. [1], the senior population will reach 2.1 billion by the year of 2050. As a person aging, the functionalities of perception and motor systems degrade in general. Therefore, compared to young persons, elderly people tend to suffer from musculoskeletal deficits, inefficient interactions with the environment, perceptual difficulties, cognitive disorders, and so forth more frequently, which can cause poor living qualities and even unexpected death. For example, a normal living environment for young people can be dangerous for elderly people. When walking on a slippery floor, a young person can effectively balance the body and avoid falling. On the other hand, an elderly person with cognitive disorders would lose the body balance and suffer from fatal falls. Another example is, that an elderly person with abnormal cognitive abilities (e.g. Alzheimer's disease) can forget the goal of his/her ongoing actions. The elderly person could forget taking medicine while preparing for sleep, or forget switching off ovens after cooking, which can heavily increase risks in daily lives.

Since the biological aging process is not stoppable and reversible by nature, our ultimate goal is to propose and utilize state-of-the-art technologies to improve the living qualities of elderly people, especially for the ones living alone. We aim at improving the technologies of ambient intelligence, so that the interaction between elderly people and the intelligent environment is as convenient as the interaction between young people and the conventional environment. We expect an intelligent environment to possess two capabilities: (1) perceiving the behaviors of elderly people and (2) performing actions to provide assistance. In this PhD thesis, we focus on the first capability. The second one is more related to robotics, mechatronics and other fields, which are out of our scope here. An intelligent perceiving system, which is able to effectively understand behaviors of elderly people, is highly expected in practice, so that healthcare persons can be informed on time when an accident occurs, forgetting the action goal can be avoid by automatic

reminding, and so on. Due to the rapid development of computer vision, especially on visual human behavior analysis, we think that enabling the environment to perceive and understand human behaviors in a non-obstructive, intelligent, and reliable manner is realizable, despite challenging.

1.1 Problem Statements and Objectives

Motivated by the demands of elderly people healthcare and our perspective on ambient intelligence, we aim at proposing state-of-the-art methods of visual human behavior analysis and applying them into various healthcare scenarios, such as fall recognition, fainting detection and analysis of fine-grained daily living actions. Since human behavior covers a large number of different types, we first propose a taxonomy, and then focus on the behavior type which is most related to elderly people healthcare. The proposed human behavior taxonomy is listed in Table 1.1 and some examples are shown in Figure 1.1.

Environment	indoor	in the wild
Number of persons	single person	multiple persons
Complexity	simple action	composite activity
Granularity	coarse-grained	fine-grained

Table 1.1: Human behavior taxonomy.

In this thesis, we focus on the scenarios of a *single person* in an *indoor* environment, in which the person can perform both simple and complex actions at different granularity levels. Such scenarios are most related to daily activities of elderly people living alone. In addition, we expect that the environment can perceive and understand human behaviors with minimal interventions to the daily living, and hence we primarily consider to use cameras, so as to capture human actions in a non-obstructive manner.

Although we have constrained the scenarios to investigate based on our focus for a number of reasons, human behavior understanding is still challenging for a number of reasons: *First*, in contrast to many datasets for action recognition, in which the video is trimmed to only contain a single type of action, the video in practice is normally

Figure 1.1: Some imagery examples of different types of human behaviors. From left to right and from top to bottom: (1) A snapshot from the PROX recording [2]. (2) Outdoor climbing (photographed by this dissertation author, i.e. Yan Zhang). (3) Preparing foods (a snapshot from the MPII cooking dataset [3]). (4) People in Tübingen, Germany (photographed by this dissertation author, Yan Zhang). (5) Simple actions (snapshots from the KTH action dataset [4]). (6) Composite activity of preparing salads. The snapshot is from the 50 Salads dataset [5], and this video is processed by the work of Zhang et al. [6]. (7) Pedestrian tracking as rigid objects [7]. This picture is from the cited manuscript © 2013 IEEE. (8) Human body joints detection and tracking by [8]. This picture is captured by the dissertation author, i.e. Yan Zhang.

untrimmed, can last for long time, and incorporates various action types. In this case, temporally locating and recognizing different actions simultaneously is an challenging task. *Second*, the human behavior can be at multiple granularity levels, but there exists no universal method of action analysis at all levels. For example, the algorithms of tracking the person as a whole and tracking the hand joints are considerably different. *Third*, it is difficult to extract reliable action features from videos due to clutters in the scene. For example, the furniture in the room can cause severe occlusions of the human body, making action inference a highly ill-posed problem. To address all these issues, we formulate our problem as **human action parsing in untrimmed videos**: Given an untrimmed video (or image sequence) with multiple consecutive actions, we aim at proposing a solution to assign each frame an action label. In this case, action recognition, temporal action segmentation, detection and other tasks are unified under one framework. Our human action parsing setting is illustrated in Fig. 1.2.

Human action parsing can be addressed by a two-stage method. In the first stage, we propose algorithms to trim the video into several disjoint segments automatically, each of which is expected to have only one action type. In the second stage, we can use off-the-shelf action recognition algorithms, e.g., [9, 10, 11], to assign each segment an action label. Since the second stage is a typical action recognition problem and has been heavily investigated in the literature, in this thesis, we focus to investigate the first stage, i.e., segmenting actions temporally from sequential data. Specifically, we propose high-efficient and unsupervised temporal segmentation methods, and verify their effectiveness for action segmentation and abnormal behavior detection via extensive experiments (see Chapter 3). Since such high-efficient and unsupervised natures enable fast processing time sequence data without labels, our proposed methods can also be used in other applications, like segmenting large-scale motion capture data and generating action proposals from unlabeled time sequences.

This two-stage method is suitable for cases, in which the input untrimmed time sequence is without annotations and off-the-shelf action recognition methods can be used. When frame-wise action annotations are available in untrimmed videos, human action parsing can be addressed by an one-stage method, i.e., performing temporal action segmentation and frame-wise labeling simultaneously in an end-to-end manner, based on deep neural

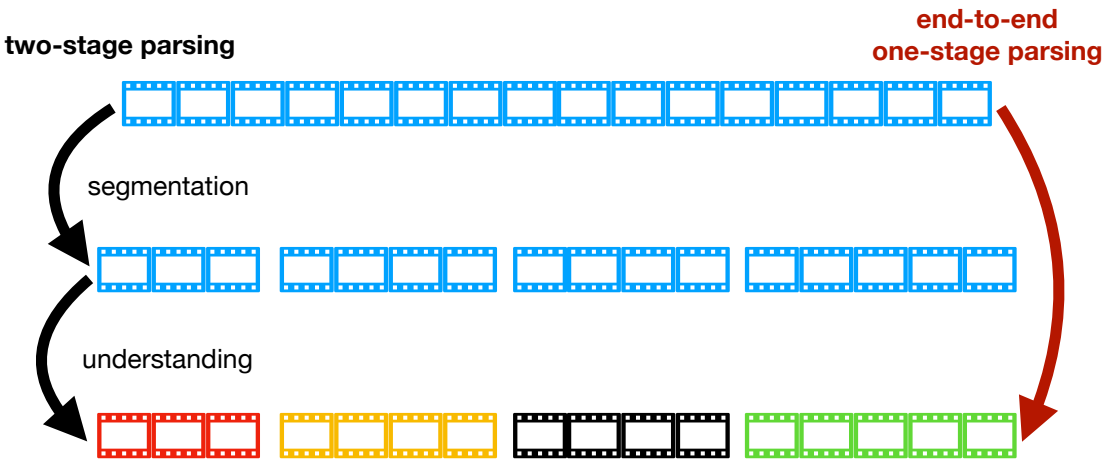

Figure 1.2: Illustration of the human action parsing problem: Given an image sequence, an action parsing method is expected to produce frame-wise action labels. For **two-stage parsing**, unsupervised temporal action segmentation and action understanding in individual segments are separately proposed. For **one-stage approach**, the segmentation procedure and the understanding procedure are inherently combined via a deep neural network, and the action parsing result is directly produced in an end-to-end fashion, given the image sequence as input.

networks. In this thesis, we train a deep convolutional neural network using videos with frame-wise action annotations, and then employ the trained network to assign each individual frame in the test video an action label. To realize fine-grained action parsing, e.g. not only recognizing cutting but also differentiating cutting tomatoes from cutting cucumbers, we propose bilinear pooling operations as network intermediate layers, and use them to extract high-order information (e.g. covariance) from data. We conduct extensive experiments, and show that our proposed methods can effectively parse fine-grained actions and outperform other relevant methods on various datasets.

1.2 Structure of The Thesis

According to our problem formulation, this thesis is structured as follows: In Chapter 2, we introduce a number of related studies towards a comprehensive review of visual human action understanding, and applications on elderly people healthcare.

In Chapter 3, we present our method for unsupervised human action parsing, which is inspired by the effectiveness of bag-of-visual-words methods [12] for action recognition. First, we propose a dynamic clustering algorithm to group visual features in an online manner. Then, regarding the dynamic clustering algorithm as a building block, we propose a framework, i.e. hierarchical dynamic clustering, in which local temporal pooling is applied to aggregate low-level visual features to high-level action patterns. We apply our methods for temporal action segmentation and fainting detection in omnidirectional videos.

In Chapter 4, we present our deep learning-based end-to-end action parsing method. We first demonstrate the spatiotemporal convolutional encoder-decoder network, and then apply it in the application of fall detection from videos. In particular, we empirically investigate the influence of multiple modalities and training schemes on fall recognition, towards understanding the inference process of the convolutional networks. Afterwards, in order to perform fine-grained action parsing, we propose two local temporal pooling methods for capture high-order information from data. The proposed bilinear pooling methods are regarded as generic intermidiate layers, and are employed in different kinds of neural works. The experimental results on various datasets indicate the effectiveness of our methods.

In Chapter 5, we conclude our work in this thesis, and provide outlooks for future work.

2Related Work

The human behavior is a term with wide meanings and can be categorized in different ways. According to the review literature on human behavior analysis and assisted living technologies, e.g. [13, 14, 15, 16], in the indoor environment, the behavior of a person can be categorized into a hierarchy, according to the granularity and the temporal duration. In this thesis, we follow the work of [14], and define four behavioral levels, namely *action primitive*, *action*, *activity* and *behavior*. At the lowest *action primitive* level, the time duration is very short and the body configuration only varies slightly, e.g. lifting the left foot during walking or putting the right arm down after waving hands. Going a level up, a time sequence of action primitives composes an *action*, such as running and sitting down. Above the *action* level, an *activity* is defined as a series of actions occurring with a specific temporal order. At the top, a behavior is defined as a set of activities, and can last for hours or even days. Healthcare solutions including sensors, algorithms and computer systems are mostly developed for each individual behavioral level, due to their diverse properties and ambiguous differentiating boundaries.

In Chapter 1, we have stated that we focus on human actions at various granularity levels. Here we further specify our focus, i.e., we focus on visual human behavior analysis, and concentrate to analyzing action primitives, actions and activities, which only last for seconds, minutes but usually less than one hour. In this case, the human movement is feasible to be recorded by cameras, and the amount of recorded data is not excessive. For long-term behavior analysis in living environments, which lasts hours, days or even months, wearable sensors, radio frequency modules, infrared sensors and other types of modalities are usually used [15, 17, 18, 19] rather than cameras. For example, deriving the sleeping cycle of a person takes weeks. Unusual changes of daily routines of elderly people are monitored via a sensor network mounted on the door, and the indoor daily mobility pattern is extracted based on recordings for days [20]. Therefore, long-term behavior analysis is out of the scope of this thesis.

In the following paragraphs, we present a review of behavior analysis methods, as well as practical solutions to various elderly people healthcare scenarios. Overall, we focus to investigate methods for the action parsing task introduced in Chapter 1.

2.1 Unsupervised Human Action Parsing

2.1.1 Tasks and Methods

Given a recording of human motion in the format of the RGB video, motion capture data, or a generic feature vector sequence, the task of human action parsing is to partition this continuous motion into several disjoint segments over time, and each segment incorporates an action primitive, which is represented by a cluster ID.

Human action parsing is related to continuous action understanding [21, 22, 23, 24] and action detection in untrimmed videos [25, 26, 27, 28, 29, 30], which are prevalent in recent years. In many cases, data annotation is not sufficient or not available, and an effective unsupervised method is highly expected. In the following, we introduce some classical and recent methods for unsupervised human action parsing.

Zhou et al. [31] proposes aligned cluster analysis (ACA) that uses a novel kernel for time series alignment. After specifying the number of clusters, as well as the minimum and maximum lengths of the action segments, both the ACA and HACA (hierarchical ACA) solve a dynamic program over the entire time sequence. Despite effectiveness, ACA and HACA are not applicable for fast and online video segmentation due to its inefficient optimization process. The work of [32] proposes an efficient motion segmentation approach, in which a novel feature bundling method is used to generate compact and robust motion representations. In particular, a generalized search radius is introduced in [32], so that the number of clusters is not needed as input. Li et al. [33] addresses motion segmentation via subspace clustering, in which a temporal Laplacian regularization method is applied to encode the temporal structure in the subspace, and then spectral clustering is applied to group actions using the projected feature vectors.

Another related line of research is the mixture model for clustering, which can be trained online. The work of [34] proposes a greedy method for training the Gaussian mixture model, but requires to specify the model complexity, e.g., the number of Gaussian components k, in advance. To overcome this, the Bayesian nonparametric models learn the value of k from the input data jointly with the observation models. For example, the Dirichlet process mixture model (DPMM) [35, 36] uses a Dirichlet process to allocate samples and calculate k, where the joint probabilities of allocations are independent of the temporal order of the input samples. The work of [37] proposes an algorithm to train DPMM by iteratively processing data batches and allowing the number of clusters varying when the model parameters are updated online.

In this thesis, we propose a dynamic clustering algorithm to group elements in the time sequence in an online manner, as well as a hierarchical dynamic clustering pipeline with temporal pooling to aggregate low-level spatiotemporal features. They have been published in [38] and [39], respectively. As investigated by systematical experiments, we show that our proposed methods outperform other state-of-the-art unsupervised human action parsing algorithms. Details of human action parsing via hierarchical dynamic clustering are demonstrated in Chapter 3.

2.1.2 Healthcare Applications

Here we introduce two healthcare applications, which are highly related to unsupervised human action parsing, namely gait analysis and abnormality detection.

Gait analysis. In the domain of healthcare and clinical diagnosis, gait is widely used as an important indication of the cognitive and motor functionalities of elderly people [40, 41, 42]. For example, the studies in [43] and [44] perform assessing and diagnosing tuning problems in patients with Parkinson, employing a hidden Markov model to explicitly model the gait cycle and partition the walking process to four phases. The work [45] performs gait analysis using accelerometer for frailty detection of elderly people. The work [46] performs person identification via analyzing the gait pattern from silhouette.

Unsupervised abnormality detection. Abnormal behaviors can be regarded as statistical outliers, given the distribution of the normal behaviors. In this case, the abnormal behavior is very similar to the novel behavior after long-term observation. One can read [14] and [47] for reviewing computer vision-based methods on abnormal behavior analysis. The work [47] presents a generic framework for abnormal behavior detection, in which the model (e.g. Gaussian mixture model, hidden Markov model and clustering structure) learns visual patterns of normal behaviors, and detects abnormalities in testing videos by evaluating the matching score and detecting the ones below a certain threshold. The work of [48] parses the video into three components, learns the dominant components and abnormal behavioral patterns gradually. The work of [49] proposes a hierarchical clustering approach to detect abnormal body configurations and abnormal actions (defined as abnormal transitions between body configurations in their work) consecutively. The work of [50] employs a force model to detect abnormal social behaviors of people crowd in the public area. The work of [38] and [51] employs hierarchical dynamic clustering method to aggregate low-level visual cues and learn action clusters to detect fainting in omnidirectional videos.

2.2 Supervised Human Action Parsing

In the following we introduce several related work about supervised action parsing in untrimmed videos, as well as relevant applications of elderly people healthcare. One can recall that the untrimmed video is defined as a video containing multiple human actions.

2.2.1 Tasks and Methods

An untrimmed video contains a sequence of consecutive actions. The task of supervised human action parsing is to assign each individual frame in the untrimmed video an action label, via a deep neural network. Therefore, action detection, action recognition, segmentation, action search and other tasks are unified within the same framework.

The work of [52] proposes to learn object and material states, for which action segmentation is performed by detecting the state transitions. The work of [29] applies a statistical language model to capture action temporal dynamics. The work of [53] proposes an Ego-ConvNet to perform temporal action segmentation, which incorporates two streams for extracting spatial features and spatiotemporal features individually from pre-defined video segments. The results are improved when combining Fisher vectors [10] from spatial and optical flow descriptors [54]. The work of [55] proposes a multi-modal bidirectional LSTM model to generate a label sequence of a video to incorporate forward and backward temporal dynamics. Lea et al. [56] proposes a conditional random field with skip connections in the temporal domain, and starting-and-ending frame priors learned via a structured support vector machine. The work of [57] proposes a multi-modal deep neural network with the similar structure of the VGG network [58]. After training and extracting frame-wise features, a temporal convolutional network and a semi-Markov conditional random field are applied to produce the final segmentation result. Based on the spatial features from [57], the work of [59] proposes two kinds of temporal convolutional networks with the encoder-decoder architecture to yield state-of-the-art performances on several fine-grained action datasets. The first network comprises layers of convolution and max pooling; the second net uses dilated temporal convolution and skipped connections to capture long-range temporal structures. Based on this work, many neural networks with sophisticated architectures are proposed in recent years, such as multi-stage dilated temporal convolution [60] and deformable temporal convolution [61].

Since the neural networks introduced above cannot capture second-order (or higher) information from features, we investigate to combine bilinear pooling and deep neural networks for fine-grained action parsing in this thesis. Bilinear pooling (or second-order pooling) is widely used in fine-grained visual classification [62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78], visual questioning answering [79, 80, 81], feature fusion and disentangling [65, 66, 82, 83, 84, 85], action recognition [86, 87, 88, 85, 84] and other tasks. In deep neural networks, bilinear pooling is mostly used only once before the classification layer, e.g. in [65, 66, 68, 69, 70, 83, 84, 75, 76, 78], or embedded within the classifier, e.g. in [73, 71].

There are several research directions regarding bilinear pooling: (1) Dimension reduction while minimizing information loss. [69, 77, 86] use tensor sketch [89] to reduce the dimension of vectorized bilinear forms. The studies of [66, 75] use parametric dimension reduction approaches, which can be learned via back-propagation. The work in [80] [74] finds a low-rank approximation of the bilinear forms, so as to convert vector outer-product in the conventional bilinear pooling into Hadamard multiplication for cheap computation. [73, 67] utilizes singular value decomposition (SVD) to select the principle components, which can increase the performance at a higher computational cost. (2) Multiple bilinear pooling layers in deep neural networks. [85] factorizes bilinear composition into consecutive matrix multiplications along different dimensions. [74] uses the low-rank approximation as in [80], and aggregates features hierarchically. (3) Methods to capture richer feature statistics, so that distributions beyond Gaussian can be represented. [90] propose a higher-order pooling scheme to extract feature co-occurrences of visual words based on linearization of a higher-order polynomial kernel. [91] applies tensor sketch to generate a compact explicit feature map up to p-th order. In spite of increasing the representativeness, more computational loads are introduced.

We propose bilinear pooling methods from two perspectives: From the *statistics* perspective, our work [92] uses the temporal encoder-decoder architecture proposed by [59]. To capture second-order statistics, the max pooling in [59] is replaced by novel bilinear pooling operations. From the *algebra* perspective, our work [93] uses low-rank tensor decomposition and random features to approximate second-order (even infinite-order) information for feature dimension and computational load reduction. Details and comparison to other state-of-the-art methods are referred to Chapter 4.

2.2.2 Healthcare Applications

For supervised human action parsing, we focus on the applications of fall detection and fine-grained action parsing in daily activity videos. Detecting falls reliably and efficiently is essential for elderly people healthcare, protecting elderly people from fatal injuries and providing assistance without delay. Additionally, fine-grained actions are very common in daily living scenarios such as cooking, cleaning and making breakfast tee, and hence an

effective algorithm for fine-grained action parsing is important for smart user interface for the people assistance purpose.

Fall detection in videos. Systematic reviews of fall recognition and detection systems can be found in [94] and [95], which cover solutions based on diverse types of sensors. For vision-based methods, a typical processing pipeline consists of background subtraction, feature extraction and classification, as presented in [96, 97] and others. Each step in this pipeline is normally hand-crafted, separately considered, and implemented based on certain heuristic rules. A frequently considered rule is that the background information is redundant for fall detection. Thus, background subtraction is performed by algorithms like training Gaussian mixture models, subspace clustering, or other sophisticated approaches [98]. Another heuristic rule is that the body shape is a pronounced feature of falling. Consequently, the silhouette of the human body [96] [99], or the shape of the foreground bounding box [97, 100], is extracted and analyzed. Nevertheless, heuristics are not always precise and comprehensive. The studies of [101] and [102] present effective fall detection solutions when considering the ground plane, indicating that the background information can be very useful.

Comparing with traditional vision-based approaches, deep learning methods enable end-to-end inference with minimal pre-processing on the input data, and the deep networks can learn representative features from the data automatically. Several studies report that deep learning methods lead to better performances in terms of action recognition [103] [104], action detection [105, 106, 107], action segmentation [59, 108], and other tasks of human behavior analysis. Their success encourages many studies of fall recognition based on deep neural networks. For example, the work of [109] employs a convolutional network with a similar architecture to the VGG16 net [58], and uses optical flow as the input modality. The work of [110] uses a PCANet to recognize falls from image sequences with the assistance of foreground detection.

When using deep convolutional networks to recognize fall in a reliable manner, it is essential to understand how deep neural networks work. For such model explanation purpose, several types of attribution maps have been proposed in the past [111, 112].

For a specific input and a target class, the attribution map has the same spatial resolution with the input, and reveals the influence of each input pixel to the probability of the target class. The work of [113] proposes a saliency map, which is computed as the derivative of the output with respect to the input. [114] proposes the integrated gradients, in which the values show the difference between the net output of a reference input (normally zero) and the net output of a sample. [115] proposes the DeepLIFT attribution measure, which can be regarded as an approximated version of integrated gradients according to [112].

Fine-grained action parsing in daily scenarios. Parsing fine-grained actions over time is important in many applications, which require understanding of subtle and precise operations in long-term periods, e.g. daily activities [116], surgical robots [117], human motion analysis [38] and animal behavior analysis in the lab [118]. Therefore, understanding fine-grained actions in videos has great potentials for our purpose of elderly people healthcare.

The work of [116] creates a set of daily fine-grained action videos recorded via an egocentric camera, and proposes an action recognition method considering both motion features and egocentric features. The work of [119] proposes a two-stream (appearance stream and motion stream) approach to perform object detection, and hand action understanding jointly. The work of [5] creates a multi-modal dataset of preparing salads, and performs fine-grained action recognition via RGBD data and accelerometer information.

Instead of designing complex network architectures for fine-grained action parsing, we focus on high-order information extraction, and propose bilinear pooling operations, which can be employed as generic intermediate layers in arbitrary deep neural networks. One can see the superior performances of the proposed bilinear pooling methods than other state-of-the-art methods in Sec. 4.4.

3Unsupervised Human Action Parsing via Hierarchical Dynamic Clustering

In this chapter, we address the problem of human action parsing in an unsupervised manner: Provided a time sequence of human motion features, we aim to partition the entire sequence into several meaningful segments, and group these segments into different clusters. Since no semantic labels are applied, each frame is assigned by a cluster ID.

An effective unsupervised human action parsing algorithm can be widely used in many applications, e.g. processing motion capture data, video summarization, detecting abnormal behaviors and so forth. In our study we propose a hierarchical dynamic clustering (HDC) pipeline (published in our work [51]), which is inspired by the *Bag-of-Visual-Words* (BoW) pipeline [120, 121, 122, 12] for supervised action recognition. A classical BoW pipeline usually comprises 5 modules, i.e., spatiotemporal feature extraction (e.g. histogram of optical flow [123] around spatiotemporal interest points [124], improve dense trajectory features [125], etc.), codebook learning, feature encoding, feature pooling and classification, in which only the last classification module is supervised and the first four are unsupervised.

In this chapter, we present how to modify the pipeline from BoW to HDC, making it suitable for unsupervised human action parsing. In particular, we have the following two novel modules:

- **Dynamic clustering.** First, we propose a dynamic clustering algorithm for codebook learning and temporal segment clustering. Compared to the traditional k-means or spectral clustering approach, our dynamic clustering method does not require to specify number of clusters in advance, has lower time complexity, and can process sequential data in a temporal *causal* manner. Thus, it is more suitable for time sequence processing, especially in cases without prior knowledge of the number of clusters, or in cases requiring online and fast processing, such as real-time fall detection in smart home.

- **Local temporal pooling.** Second, we propose local temporal pooling schemes to replace the global pooling in the standard BoW pipeline. Temporal pooling is essential, since the input spatiotemporal features, which are normally extracted from individual time instants or very-short periods, are too local to represent actions. The results directly from dynamic clustering can be severely over-segmented. The local temporal pooling is to aggregate *encoded* spatiotemporal features within local time windows. To determine these time windows and extracting action patterns from individual time windows, we propose a kernelized cut-based pooling scheme and a motion energy-based pooling scheme. Compared to the classical time sliding window-based pooling, which normally uses a pre-defined window size, our kernelized cut-based scheme and motion energy-based scheme are able to produce data-adaptive time windows, so that the detected temporal boundaries are more accurate and the extracted action patterns are more representative.

Tab. 3.1 presents the comparison between the BoW and HDC pipelines, and Fig. 3.1 illustrates our HDC pipeline. Same with BoW, our HDC framework is generic and can take various types of spatiotemporal features as input. These features can be either frame-wise, or representing actions in a short video snippet. To demonstrate our HDC framework in a generic manner, we don't mention any specific input features here. One can see experiments with various input features in Sec. 3.6.1, in which the employed input features are frame-wise body joint rotations, output from a deep neural network, Fisher vectors from hand-crafted trajectory features, and so forth.

	feature extraction	codebook learning	feature encoding	feature pooling	labeling
BoW	same	k-means/GMM	same	global	classification
HDC		DC		local temporal	DC

Table 3.1: Comparison between bag-of-visual-words (BoW) and hierarchical dynamic clustering (HDC). Here DC stands for dynamic clustering.

In the rest of this chapter, we first present some preliminaries about BoW, and show an example of how a BoW method works. Then, we successively introduce the modules in our HDC pipeline, i.e. *dynamic clustering*, *feature encoding*, *local temporal pooling* and *temporal segment clustering*. Afterwards, we show the experiments on temporal action segmentation in videos and fainting detection in omnidirectional videos. One can see

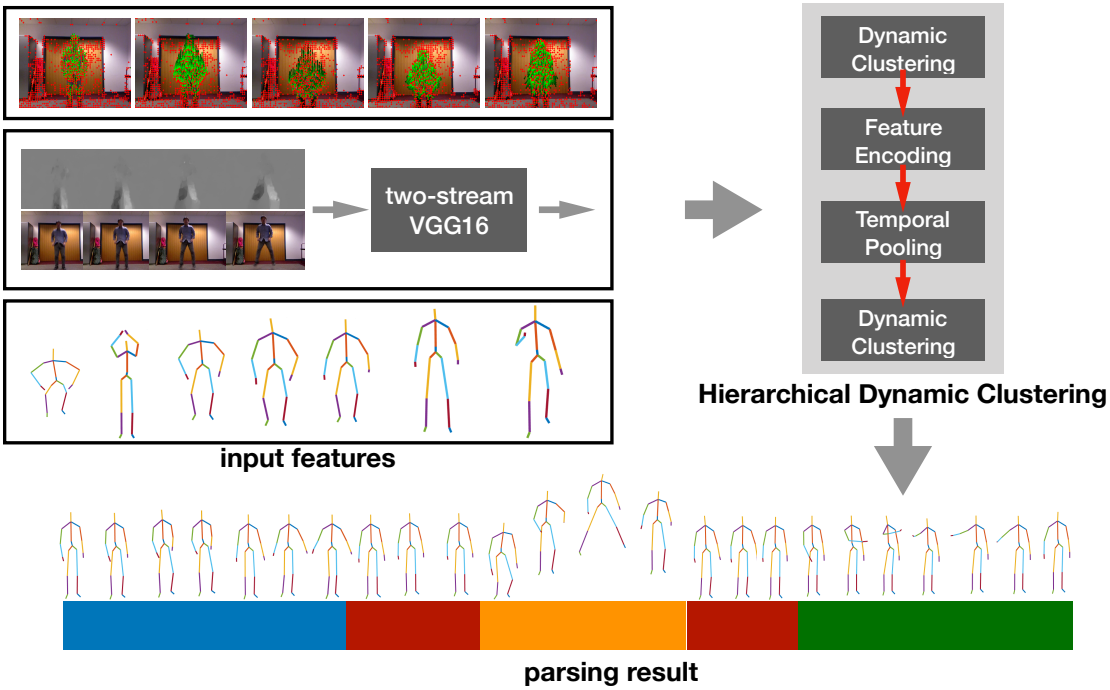

Figure 3.1: Illustration of our hierarchical dynamic clustering (HDC) method for human action parsing. The input features are generic, and can be IDT+FV [125], outputs of convolutional networks [103] or motion capture data. After raw feature extraction, the following 4 components in HDC are applied, and produce a 1-D piece-wise constant sequence indicating the human parsing result. In this figure, colors denote different types of actions.

that our HDC method outperforms other state-of-the-art unsupervised methods in terms of quality and speed. In the end, as a conclusion we discuss the pros and cons of our HDC method for unsupervised human action parsing, and give some tricks of how to use it in practice.

3.1 Preliminaries

Here we present some preliminary concepts of the *bag-of-visual-words* (BoW) pipeline, and a few examples of how the BoW method works.

Before deep learning, the BoW framework is widely used in image classification [126] and action recognition [12]. The aim is to derive a *single* compact representation of an image or a video, via aggregating a set of local features (e.g. SIFTs [127]) in a bottom-up manner. This bottom-up feature aggregation scheme can be regarded as a simulation of how humans perceive and understand things in the world. For example, when seeing an image, the V1 cell in the human vision system extracts image edges. After encoding and aggregating all the edges, the visual cortex produces semantic meanings of this image. When hearing a talk, a person hears the tone of each word sequentially. These tones are then aggregated in the brain, so that the person can understand the meaning in the talk. Without such bottom-up feature aggregation, a person cannot understand things only based on low-level features. A person cannot understand what is in the image, only by seeing an image edge. A person cannot understand what another person is talking, only by hearing the tone of a language letter.

A standard BoW framework comprises 5 steps, i.e., (1) local feature extraction, (2) codebook learning, (3) feature encoding, (4) feature global pooling and (5) classification. Such bottom-up feature aggregation scheme shares some similarities with deep neural networks, such as the VGG network [58] and the C3D network [128]. However, rather than learning end-to-end in deep neural networks, the first 4 steps in BoW are unsupervised and only the last classification step is supervised.

Let's take a toy example of how to use BoW to extract compact representations of 3 walking paths on a 2D ground plane. We assume that the first walking path W_1 has N_1 frames, the second walking path W_2 has N_2 frames, and the third walking path W_3 has N_3 frames, respectively.

(1) Feature extraction. From each frame of each walking path, we extract the 2D location as the local feature. Therefore, the first walking path is represented by a set of N_1 locations, the second walking path is represented by a set of N_2 locations, and the third walking path is represented by N_3 locations, respectively.

(2) Codebook learning. Here we use Kmeans to cluster all $N_1 + N_2 + N_3$ locations in the three sets to K clusters, i.e. $\{\boldsymbol{c}_1, \boldsymbol{c}_2, ..., \boldsymbol{c}_k\}$. The K clusters are regarded as the codebook.

(3) Feature encoding. For simplicity, here we use the hard assignment scheme [12] as an example. For a 2D location $\boldsymbol{x}$ in a walking path, we compare the distance between $\boldsymbol{x}$ and these cluster centers in the codebook, and then assign the ID of the closest cluster to $\boldsymbol{x}$, i.e., $i = \arg\min_k \|\boldsymbol{x} - \boldsymbol{c}_k\|_2^2$. Next, we encode the location $\boldsymbol{x}$ to a *K-dimensional* one-hot vector, in which the k-th element is 1 and other elements are 0. We denote this K-dimensional one-hot vector as $\phi(\boldsymbol{x})$.

(4) Feature global pooling. After feature encoding, a walking path can be represented by a set of encoded features, i.e., $\{\phi(\boldsymbol{x}_1), \phi(\boldsymbol{x}_2), ...\}$. To derive the compact representation for each individual walking path, we perform global pooling to aggregate these encoded local features. For simplicity, we take average pooling as an example. In this case, the compact representation of the first walking path is $\sum_{\boldsymbol{x} \in W_1} \phi(\boldsymbol{x})/N_1$, the compact representation of the second walking path is $\sum_{\boldsymbol{x} \in W_2} \phi(\boldsymbol{x})/N_2$, and the compact representation of the third walking path is $\sum_{\boldsymbol{x} \in W_3} \phi(\boldsymbol{x})/N_3$. According to the investigation in [12], an additional normalization step, e.g. l1 and l2 normalization, after the global pooling is beneficial.

(5) Classification. After the four steps above, we can get a compact representation for each individual walking path. If each walking path has a ground truth label, for example, the first walking path is “walking” and the second walking path is “running”, then one can train a classifier (e.g. support vector machine [129]) for classification. During the testing phase, we following the same four step to derive the compact representations of test walking paths, and then use the trained classifier to make predictions. One should note that encoding features of test samples should be based on the codebook learned from training samples.

The above five steps are performed consecutively. Advanced methods to improve individual steps can boost the overall performance. For example, Fisher vector encoding [10] yields better image recognition results than the standard BoW method, by employing Gaussian mixture models to learn the codebook and incorporating second-order statistics during feature encoding.

Our HDC method has three major differences from the standard BoW method: (1) We use a novel dynamic clustering method to learn the codebook. (2) Rather than performing global pooling to encode an entire time sequence into a single compact representation, we perform local temporal pooling to partition an entire time sequence into several segments, and extract a compact representation from each individual segment. (3) Rather than training a classifier supervisedly, we perform dynamic clustering on the sequence of compact representations, so as to assign each temporal segment a cluster ID. Therefore, our HDC method is fully unsupervised.

3.2 Dynamic Clustering

In this section, we present our dynamic clustering algorithm, which is specifically designed to cluster frames in time sequences. We have published our dynamic clustering method in [38]. To focus on algorithm demonstration, we do not specify details of input features. One can see Sec. 3.6.1 for experiments with various feature types, such as frame-wise human body joint locations and Fisher vectors extracted from dense trajectories.

Given a time sequence of generic feature vectors, our dynamic clustering algorithm assigns each individual entry in the sequence into a cluster, and hence can partition the entire sequence into disjoint segments. Formally, we assume that we are given a sequence of vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t, \ldots$ as input, in which every $\boldsymbol{x}_i \in \mathbb{R}^d$ is the feature vector at time i. To make our approach applicable in realistic scenarios, we do not make additional assumptions on the input feature vectors, in the sense that each new arriving input $\boldsymbol{x}_i$ could be either a feature vector corresponding to a single frame or a video snippet (e.g. 50-frame video clip).

The design of our algorithm is based on the fact that the feature vectors representing the same human actions are similar to each other. Therefore, in a time sequence, a new arriving action feature is grouped with "similar features" observed before, and partitioned from "dissimilar" features observed before. To achieve this goal, our dynamic clustering algorithm have two steps, i.e., an initialization step and an updating step. We give conceptual descriptions here, and then demonstrate their details in Sec. 3.2.1.

The initialization step aims to quantify the sense of "similar". Our algorithm first computes the basic statistics of the feature vectors in the time sequences. Specifically, we build a fully connected graph G from the first few feature vectors (e.g. the features arriving in the first 1 second), and run spectral clustering on G. In contrast to conventional spectral clustering algorithms that only cluster the vertices of G, our algorithm computes the cluster centers $\{\mathbf{c}_i\}_{i=1}^k$ to measure *how far clusters are separated from each other*, as well as the diagonal matrices $\{\mathbf{\Sigma}_i\}_{i=1}^k$ to measure *how concentrated all the feature vectors within each cluster are around the center*. We define the cluster set $\mathcal{C}$ as the combination of $\{\mathbf{c}_i\}_{i=1}^k$ and $\{\mathbf{\Sigma}_i\}_{i=1}^k$.

After obtaining the basic feature statistics, our algorithm enters the updating phase, and runs in an online fashion. Specifically, for every arriving feature vector $\boldsymbol{x}_t$, the algorithm computes the distance between $\boldsymbol{x}_t$ and its closest center of all the observed clusters $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_k$, i.e., $\mathsf{dist}(\boldsymbol{x}_t, \mathcal{C}) = \min_{1 \le i \le k} \|\boldsymbol{x}_t - \boldsymbol{c}_i\|^2$. Based on $\mathsf{dist}(\boldsymbol{x}_t, \mathcal{C})$, and matrices $\{\mathbf{\Sigma}_i\}$, the algorithm decides whether to put $\boldsymbol{x}_t$ into its closest cluster, or generates a new cluster to host $\boldsymbol{x}_t$. One can note that the runtime of this step is only proportional to the number of currently observed clusters and the dimension of the feature vectors, which is independent of the length of the video stream. Hence, the update time for every arriving feature vector can be accomplished in $O(1)$ for most applications.

3.2.1 Algorithm

In the following, we demonstrate details of the initialization step and the online updating step, respectively:

Initialization step. The input of the initialization step are the initial ℓ feature vectors $\mathbf{x}_1, \ldots, \mathbf{x}_\ell$ from a video stream. The algorithm first constructs a fully connected similarity graph $G = (V, E)$ of ℓ vertices, denoted by $v_1, \ldots, v_\ell$, where each v_i corresponds to a vector $\boldsymbol{x}_i$ and the weight of the edge between v_i and v_j is defined as $w(v_i, v_j) = \exp\left(-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2/\sigma\right)$ with a parameter σ, which quantifies the similarity between v_i and v_j. This graph G is represented by the normalized Laplacian matrix, which is defined by $\mathcal{L} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$, where $\mathbf{I} \in \mathbb{R}^{\ell\times\ell}$ is the identity matrix, $\mathbf{D} \in \mathbb{R}^{\ell\times\ell}$ is the diagonal matrix defined by $\mathbf{D}_{i,i} = \sum_j w(v_i, v_j)$, and $\mathbf{A} \in \mathbb{R}^{\ell\times\ell}$ is the adjacency matrix of G. We then compute the eigenvalues $\lambda_1 \leq \ldots \leq \lambda_\ell$ of $\mathcal{L}$, and use the eigengap heuristic introduced in [130]. Specifically, in our implementation, we use the smallest value k with the largest absolute difference between λ_{k+1} and λ_k to determine the number of clusters. This method is widely used for determining the number of clusters in practice, see [131] for theoretical explanation and [130] for detailed discussion.

After computing the value of k, the algorithm runs k-means for the initial ℓ feature vectors $\mathbf{x}_1, \ldots, \mathbf{x}_\ell$. In addition to the k clusters produced by k-means, the following quantities are computed: (1) Centres $\{\boldsymbol{c}_i\}_{i=1}^k$ for these k clusters; (2) covariance matrices $\{\boldsymbol{\Sigma}_i\}_{i=1}^k$, where $\boldsymbol{\Sigma}_i \triangleq \mathbf{diag}\left(\left(\sigma_1^{(i)}\right)^2, \ldots, \left(\sigma_d^{(i)}\right)^2\right)$ and $\sigma_j^{(i)}$ is defined as the ℓ_2-distance of the jth coordinate between all the feature vectors in the ith cluster and their center; (3) radii $\{r_i\}_{i=1}^k$ for the k clusters and defined by $r_i \triangleq \operatorname{Tr}\boldsymbol{\Sigma}_i$. Moreover, we define a minimum area of the cluster δ_r with $\delta_r \triangleq c \cdot d$, which is similar to the search radius introduced in [32]. Here d denotes the dimension of the input feature vector to dynamic clustering, and the constant c is a hyper-parameter, which can be viewed as the uncertainty estimate of each dimension and is empirically determined depending on the use.

Online evolution step. The second step of our algorithm is the online clustering based on the statistical information of the currently observed clusters. Specifically, for every arriving $\boldsymbol{x}_t$, the algorithm computes the Euclidean distance between $\boldsymbol{x}_t$ and its closest center, i.e., $\mathrm{dist}(\boldsymbol{x}_t, \mathcal{C}) = \min_{1\leq i\leq k} \|\boldsymbol{x}_t - \boldsymbol{c}_i\|^2$. Depending on this distance, the algorithm either creates a new cluster consisting of a single point $\boldsymbol{x}_t$, or adds $\boldsymbol{x}_t$ to its closest cluster and updates the corresponding $\boldsymbol{c}_i$ and $\boldsymbol{\Sigma}_i$. The formal description is presented in Algorithm 1. Here, $\circ$ denotes the Hadamard product operation. Notice that, for the task

of online segmentation, a transitional point is detected if the current observed $\boldsymbol{x}_t$ and $\boldsymbol{x}_{t-1}$ belong to different clusters.

Algorithm 1 Online evolution step

$\text{dist}(\boldsymbol{x}_t, \mathcal{C}) = \min_{1 \leq i \leq k} \|\boldsymbol{x}_t - \boldsymbol{c}_i\|^2$.
$i^\star = \arg\min_{1 \leq i \leq k} \|\boldsymbol{x}_t - \boldsymbol{c}_i\|^2$.
if $\text{dist}(\boldsymbol{x}_t, \mathcal{C}) \geq \max\{r_{i^\star}, \delta_r\}$ **then** ▷ adding a new cluster
$k \leftarrow k + 1$; $S_k \leftarrow \{\boldsymbol{x}_t\}$;
$\mathbf{c}_k \leftarrow \boldsymbol{x}_t$; $M_k \leftarrow \boldsymbol{x}_t \circ \boldsymbol{x}_t$;
$\Sigma_k \leftarrow (0, 0, ..., 0)^T$.
else ▷ updating the closest cluster
$\mathbf{c}_{i^\star} \leftarrow \mathbf{c}_{i^\star} + (\mathbf{x}_t - \mathbf{c}_{i^\star}) \cdot |S_{i^\star}|^{-1}$;
$M_{i^\star} \leftarrow (M_{i^\star} \cdot |S_{i^\star}| + \boldsymbol{x}_t \circ \boldsymbol{x}_t) \,/\, (|S_{i^\star}| + 1)$;
$\boldsymbol{\Sigma}_{i^\star} \leftarrow M_{i^\star} - \mathbf{c}_{i^\star} \circ \mathbf{c}_{i^\star}$;
$r_{i^\star} \leftarrow \text{Tr}\, \boldsymbol{\Sigma}_{i^\star}$; $S_{i^\star} \leftarrow S_{i^\star} + \{\boldsymbol{x}_t\}$.

Determining the threshold hyper-parameter. Although the dynamic clustering algorithm does not require to specify number of clusters in advance, the value of δ_r can manipulate the number of clusters. When δ_r is too large, then all elements in the input feature sequence are grouped into one cluster. When δ_r is too small (e.g. close to 0), then each element in the input feature sequence becomes a cluster center. One can note that such threshold value has no direct correspondence to the number of clusters like in k-means. Therefore, in some applications with prior knowledge of cluster numbers, one probably needs validation data to search a value of δ_r according to the prior knowledge of number of clusters. Without prior knowledge of number of clusters, empirically one can first run spectral clustering on the validation data to determine the number clusters using the eigengap heuristic [130]. Then the value of δ_r can be regarded as half of the minimum distance between two clusters. Additionally, the threshold δ_r can explicitly represent the minimum distance of two different clusters in the feature space, and hence our dynamic clustering algorithm is well suitable for applications with the prior knowledge like “at least how far two points are belong to different clusters”. For example, transition from the “sitting” pose to the “standing” pose of a human body can cause a gap between the body joint rotations. In this case, the value of δ_r can be determined as the half of the

distance between the two joint rotation features, e.g. the concatenation of individual joint 3D rotations relative to the pelvis.

3.2.2 Runtime Analysis

In the initialization step, we first need to build the similarity graph G based on ℓ feature vectors, each of which has dimension d. This takes $O(\ell^2 \cdot d)$ time. After that, computing all the eigenvalues of the graph Laplacian $\mathcal{L}$ takes $O(\ell^3)$ time, and $O(\ell \cdot k)$ time is needed for writing down the spectral embedding of all the vertices. Since the number of clusters $k = O(1)$ for most applications, this initialization step takes $O\left(\ell^2 \cdot d + \ell^3\right)$ time.

In the online evolution step, computing the distance between every arriving $\boldsymbol{x}_t$ and its closest center takes time $O(k \cdot d)$, where k is the number of clusters at time t. After that, $O(d)$ time suffices to update all the quantities maintained by the algorithm. Hence, the update time for each arriving feature vector is $O(k \cdot d)$.

3.2.3 Qualitative Results

To qualitatively show that dynamic clustering outperforms k-means and spectral clustering for high-level action pattern grouping, we perform a qualitative experiment. Specifically, we estimate the improved dense trajectories, and derive Fisher vectors [125] of the *drinkwaterS1R1* video in the **RADL** dataset [133], which consists of three actions (i.e., *fetching water*, *pouring water*, and *drinking water*). The Gaussian mixture model has 32 components, and is trained from the videos *drinkwaterS1R2* and *drinkwaterS1R3*. The Fisher vectors have 12,288 dimensions, and are extracted using a sliding window of 50-frame length and 1-frame stride. We apply PCA to reduce the dimension of vectors to 50 before running the clustering algorithms.

The results are shown in Figure 3.2. The sequences of the high-dimensional feature vectors are visualized by t-SNE [132], which can visualize high-dimensional data in a low-dimensional space. One notes that the low-dimensional result of t-SNE preserves the topology (or distance relations) in the original high-dimensional space. However, the original shape is not preserved. A high-dimensional straight line may become to a curve

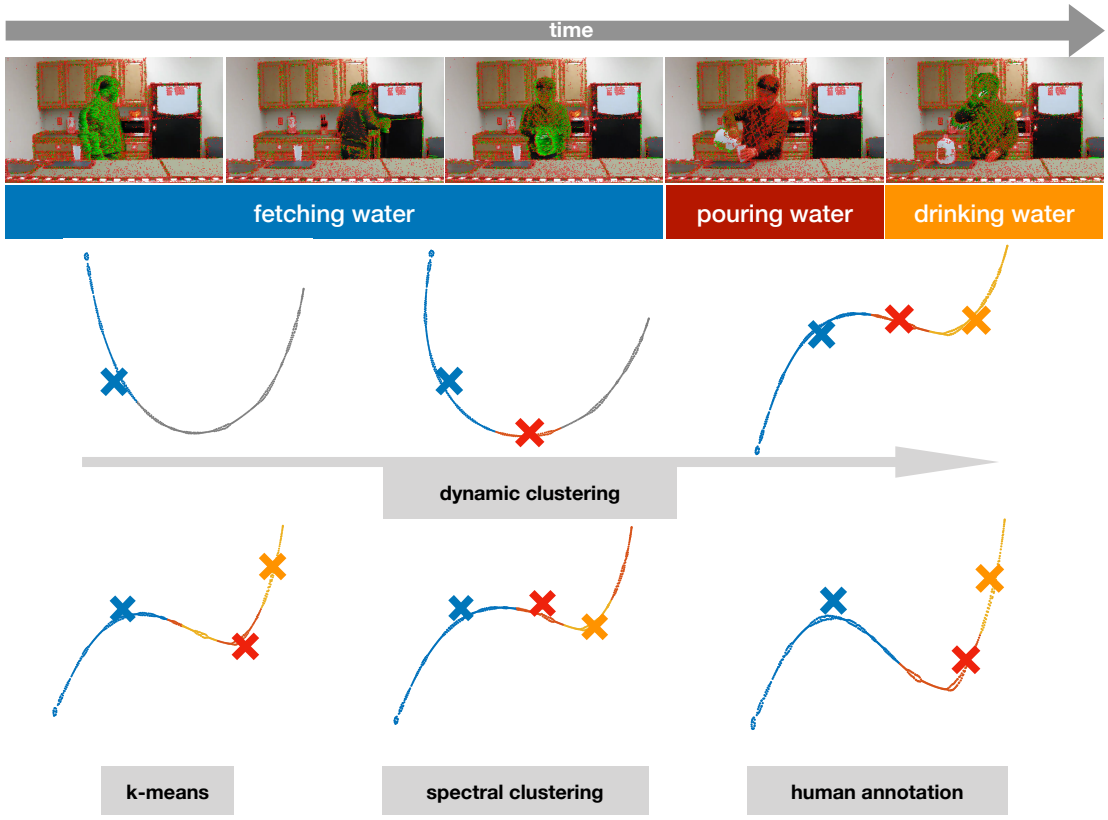

Figure 3.2: Qualitative comparison of different clustering methods. The first row shows 5 frames of a video with 3 consecutive actions. The extracted dense trajectories [125] are also rendered. The second row shows how the dynamic clustering gradually assigns cluster IDs to video frames (denoted by colors), and creates new clusters (denoted by crosses), visualized by t-SNE [132]. The third row shows the results of k-means, spectral clustering and human annotation for comparison. Thus, one can compare the three curves in the third row with the last curve in the second row, which indicates the final result of dynamic clustering. We use blue, red and yellow to denote *fetching water*, *pouring water* and *drinking water* when conducting human annotation, and denote the cluster ID 0, 1 and 2 when running clustering algorithms.

in the 2D space. Also, t-SNE is based on nonlinear optimization, and the initial values are set randomly at each run. Therefore, t-SNE can generate visualization results with different shapes, given the same high-dimensional input data. All results are valid for the visualization purpose.

From these results, one can see that the dynamic clustering algorithm is able to generate and update cluster structures in an online fashion, leading to comparable results with the human annotation. On the other hand, the k-means and the spectral clustering algorithms falsely parse *pouring water* and *drinking water* actions into non-consecutive segments, so the performance is inferior to the proposed dynamic clustering algorithm.

3.2.4 Further Discussion

Comparison with online k-means. Online k-means clustering has received considerable attention in theoretical computer science and machine learning, and it shares some similarities with our dynamic clustering method. To our knowledge, the most recent version of online k-means is proposed by Liberty et al. in [134]. Compared to this version, our proposed dynamic clustering algorithm has two essential differences:

First, it has been observed in [134] that the online k-means algorithm with bounded approximation guarantee has to generate strictly more than k clusters. On the other hand, our dynamic clustering method does not have such constraint. In fact, the threshold hyper-parameter δ_r can be regarded as an estimated measure of the feature space, and play the role penalizing of the cluster numbers. According to [135, Algorithm 1, Eq. 1], our method can be regarded as an algorithm to solve

$$\min_{\{S_c\}_{c=1}^{k}} \sum_{c=1}^{k} \sum_{\boldsymbol{x}\in S_c} \|\boldsymbol{x}-\boldsymbol{\mu}_c\|_2^2 + \delta_r \cdot k \tag{3.1}$$

$$\text{subject to } \boldsymbol{\mu}_c = \frac{1}{|S_c|} \cdot \sum_{\boldsymbol{x}\in S_c} \boldsymbol{x}, \tag{3.2}$$

in which $\boldsymbol{\mu}_c$ is the center of the cluster S_c, k denotes the total number of clusters, δ_r is the threshold in our dynamic clustering algorithm, and the operation $|\cdot|$ denotes extracting the number of samples in a cluster. One can see that a large value of δ_r can penalize the number of clusters, and decreasing the value of δ_r can increase the number of clusters.

The online k-means algorithm has a hyper-parameter k, namely the number of clusters, while our dynamic clustering algorithm has a hyper-parameter δ, which can indicate the

proximity of different actions. In some cases, for example, an online non-stopping video streaming that captures a single person behavior in a living room, it is not straightforward to determine how many different actions will occur in future, but it might be easier to specify the proximity of different actions. In this case, our dynamic clustering method is more suitable to use.

Second, the online k-means in [134] can lead to a local minimum with a large cost value. Assume all the data points are in $\mathbb{R}$, $k = 2$, and the first two arriving points are $\boldsymbol{x}_1 = 0$ and $\boldsymbol{x}_2 = 1$, respectively. At this point any algorithm will put $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ into two clusters with total cost 0, since otherwise putting them into a single cluster will increase the cost value from 0 to $1/2$. However, after an algorithm assigns $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ to different clusters, the third arriving point could be $\boldsymbol{x}_3 = c$ for an arbitrary large value c. Since the online k-means algorithm only generates a new cluster by probability, it can put $\boldsymbol{x}_3$ into one of the two existing clusters, which makes the total cost as least $\mathcal{O}(c)$ in contrast to the optimal cost value $1/2$. It is straightforward to generalize this example to the case of $k \geq 3$ clusters in high-dimensional space. On the other hand, our dynamic clustering algorithm produces deterministic solutions, after the input time sequence and the hyper-parameters of the algorithm are fixed. Therefore, the dynamic clustering method will create a new cluster for $\boldsymbol{x}_3$ in this case.

Why is dynamic clustering suitable for time series processing? To explain, we first look at the task of clustering feature vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathbb{R}^d$ into 2 clusters in an action sequence, e.g. "A→B→A", using the conventional k-means or spectral clustering methods. In the assignment step, each point is assigned to a cluster according to the closest distance, and the temporal order and temporal regularity is totally ignored. This fact can for example cause over-segmenting the continuous action "B", if it contains some body poses similar to poses in the action "A". On the other hand, our dynamic clustering processes arriving features in an online and causal fashion, updating the cluster structure only based on the current arriving feature. According to the temporal regularity of natural human motions, a time sequence of action features tends to be temporally smooth, and our dynamic clustering algorithm will receive many points around a cluster center, before receiving points from another cluster. Therefore, when encountering the ambiguous

poses in “B”, our dynamic clustering algorithm tends to assign these poses to “B”, since the cluster of “A” is still growing and far away. Fig. 3.2 shows such advantages of dynamic clustering, compared to conventional k-means and spectral clustering.

3.3 Feature Encoding

Here we present how to encode the raw features with the learned codebook. Following the standard bag-of-the-words pipeline, we use the *soft-assignment* method due to its simplicity, high efficiency and effectiveness [12]. Specifically, given a feature vector $\boldsymbol{x} \in \mathbb{R}^d$ and a set of cluster centers $\{\boldsymbol{c}_k\}_{k=1}^K$ with $\boldsymbol{c}_k \in \mathbb{R}^d$, the vector encoding is done by computing the distance between the feature vector and the clusters. In this case, the encoded feature vector $\phi(\boldsymbol{x})$ is K-dimensional, and each entry is given by

$$\boldsymbol{\phi}(\boldsymbol{x})_k = \frac{exp(-\lambda|\boldsymbol{x} - \boldsymbol{c}_k|^2)}{\sum_k exp(-\lambda|\boldsymbol{x} - \boldsymbol{c}_k|^2)} \tag{3.3}$$

for $k = 1, 2, ..., K$. Hence, all entries of the encoded feature vector $\boldsymbol{\phi}(\boldsymbol{x})$ are positive. One can note that a larger λ can make $\boldsymbol{\phi}$ more sparse. In the extreme case, $\boldsymbol{\phi}$ becomes to a one-hot vector. A smaller λ can make $\boldsymbol{\phi}$ more smooth. In the extreme case, $\boldsymbol{\phi}$ becomes to a constant vector, which does not contain information. In our study, we empirically set $\lambda = 0.1$ according to [136], so as to balance the sparseness and representativeness of the encoded feature vector $\boldsymbol{\phi}$. There exist a number of studies on feature encoding methods, which can be referred to [12]. More advanced feature encoding methods are out of the scope of our investigation.

3.4 Local Temporal Pooling

As presented in Sec. 3.1, the conventional BoW pipeline for action recognition, i.e., assigning an action label to the entire video, requires a global pooling operation to aggregate encoded local spatiotemporal features to a global action pattern, which is a compact representation of the action in the entire video. Aiming to perform unsupervised

temporal segmentation in untrimmed videos, i.e., assigning cluster IDs to individual temporal segments, we perform local temporal pooling, in order to aggregate encoded spatiotemporal local features in individual time windows to compact high-level action representations.

Such feature aggregation step is essential. In many cases, the input spatiotemporal feature is too local to represent an action. For example, if each feature vector in the input time sequence represents the body pose in each individual frame, this kind of feature vector covers too short time to represent a long-term action, e.g. running, which spans tens of frames and consists of multiple body poses. Therefore, directly applying dynamic clustering to partition the input sequence can lead to over-segmentation. The resultant cluster IDs assigned to individual frames cannot represent actions.

Our local temporal pooling can be divided into two consecutive problems, i.e., **temporal boundary localization** to determine time windows, and **feature aggregation** in individual temporal windows. One can see that determining time windows and aggregating features are "chicken-and-egg" problems: First, it is obvious that high-quality time windows are essential to extract representative action patterns. For example, provided an inaccurate time window, e.g. a time window containing half cycle of walking and then falling, the aggregated action pattern from this time window has mixed actions, and hence is not able to effectively represent either walking or falling. Second, a high-performance feature aggregation method can produce high-quality action patterns, which are easy to segregate, and are favorable for localizing temporal boundaries to determine time windows. In our study, we aggregate encoded local features via averaging. More advanced aggregation schemes are out of the current scope.

According to these natures, in the following we first present the sliding window-based pooling as the baseline, and then present our proposed kernelized cut-based pooling and motion energy-based pooling. To focus on algorithm demonstration, we do not specify the feature types here. One can see experiments in Sec. 3.6.1 with various types of input features, such as Fisher vectors from estimated dense trajectories, output from deep neural networks and frame-wise body skeleton locations.

3.4.1 Sliding Window-Based Pooling

Sliding window-based method is widely used in action detection in videos, e.g. [137]. Here, we use a sliding time window on the encoded feature vector sequence, which is derived via our codebook learning and our feature encoding approaches stated before. However, due to large variations in the temporal scales of different human motions, a unique pre-defined time window size can easily either break a coherent motion into several segments or falsely merge different motions into one, and hence often produces poor high-level action patterns as illustrated in Figure 3.3.

3.4.2 Kernelized Cut-Based Pooling

As mentioned before, determining time windows and extracting representative action patterns from individual time windows are “chicken-and-egg” problems, and suboptimal time windows from the sliding window scheme can degrade the representativeness of resultant action patterns. For example, if the time window covers both “sitting” and “walking”, then the action pattern derived from this time window is not representative for either action, no matter how advanced the feature aggregation method is. To overcome this issue, we propose a kernelized cut-based pooling method, which *iteratively* performing temporal boundary localization and feature aggregation to make patterns of different actions more distinguishable. The effectiveness of such iterative scheme is also reported by [138]. In our study, we conduct two iterations, since more iterations do not offer significant improvements, but make the computation much slower due to the graph cut. Specifically, such two iterations are as follows:

- In the first iteration, we run the sliding window-based pooling, for which the time window size is T (its value is specified later) and the sliding stride is 1 frame. Namely, we run the sliding window scheme to obtain time windows, and from each time window we aggregate encode local features via averaging. As a result, we can obtain a sequence of action patterns.
- In the second iteration, we build a fully connected graph according to the action patterns obtained in the first iteration, and then use a graph cut method to determine

new disjoint segments. Next, we aggregate the encoded input raw features by averaging within each individual temporal segment. One notes that the second feature aggregation is based on the original encoded input features, rather than the action patterns from the first iteration. The reason is that averaging the action patterns, which are obtained by averaging before, will make the results over-smooth, and degrade their capabilities of action representation.

Here we present how to create the fully connected graph and conduct graph cut. Referring to [139, 33, 140], an essential question is how to design a kernel function to specify the edge weights. Inspired by [139], we propose a kernel function to measure feature similarity with consideration of local temporal structures. Given a set of features $\{\boldsymbol{x}_i\}$, the kernel function $k(\cdot,\cdot)$ is given by

$$k(\boldsymbol{x}_i,\boldsymbol{x}_j) = k_s(\boldsymbol{x}_i,\boldsymbol{x}_j)\cdot k_t(\boldsymbol{x}_i,\boldsymbol{x}_j) = \exp(-\alpha\|\boldsymbol{x}_i-\boldsymbol{x}_j\|_2^2)\cdot\frac{\langle\bar{\boldsymbol{x}}_i,\bar{\boldsymbol{x}}_j\rangle}{\|\bar{\boldsymbol{x}}_i\|_2\cdot\|\bar{\boldsymbol{x}}_j\|_2} \tag{3.4}$$

where $\bar{\boldsymbol{x}}_i = \frac{1}{|\mathcal{N}_i|}\cdot\sum_{k\in\mathcal{N}_i}\boldsymbol{x}_k$, $\mathcal{N}_i$ denotes a temporal range centered at $\boldsymbol{x}_i$ and α is a positive constant. In our trials, we find that $|\mathcal{N}|\approx 1.5T$ (1.5 times of the dense sliding window size) empirically yield good results on different datasets. In the kernel function, the first component k_s measures the similarity of two frames straightforwardly; the second component k_t incorporates temporal information and hence can differentiate two identical features with different temporally local statistics. In addition, in our case the set $\{\boldsymbol{x}_i\}$ is obtained by the soft-assignment encoding [12] and hence is located within the domain of positive real numbers, leading to the fact that $k(\cdot,\cdot)$ is positive-definite and its co-domain is non-negative. In our empirical trials, our proposed kernel performs better than the one in [139].

Given the kernel function, one can perform two types of cut: sequential cut as in [139], and batch cut as in [33, 140]. In the sequential cut, the number of cuts are highly influenced by the temporal neighbourhood size $|\mathcal{N}|$: The shorter the neighbourhood size is, the more cuts are produced. One can note that a high recall value can be achieved in the result of human action parsing, if the number of cuts is large. In the batch cut, one is required to specify the number of action clusters in advance. Similar to specifying the neighbourhood size in the sequential cut, a larger number of clusters will result in a

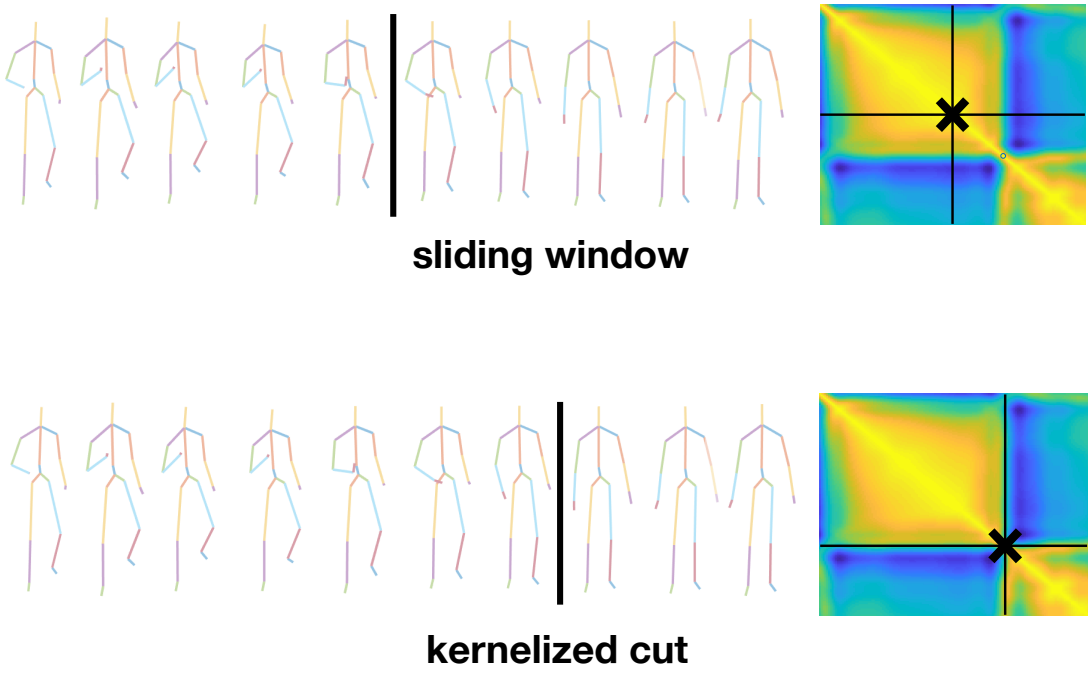

Figure 3.3: Comparison between sliding window-based pooling and kernelized cut-based pooling with the kernel function Eq. (3.4). From left to right in each pooling method: (1) A sequence of body skeletons represented by the concatenation of 3D joint locations, in which the vertical bar indicates the detected segment boundary. (2) The similarity matrix computed by Eq. (3.4), ranging from 0 (blue) to 1 (yellow), in which the cut position is shown.

larger number of cuts. Nevertheless, a smaller number of clusters can also lead to large number of cuts in certain cases.

As a qualitative result, Fig. 3.3 illustrates the benefit of a kernelized cut over the sliding window-based pooling with a pre-fixed window size. According to the skeleton sequence, which is represented by the concatenation of 3D joint locations, one can see that the kernelized cut method detects a segment boundary at a more meaningful place. According to the similarity matrix shown in Fig. 3.3, one can see that the kernelized cut finds a cut that locally maximizes intra-segment similarity and minimizes inter-segment similarity. Empirically, setting the time window size T to the frame rate is without loss of generality, and yields good results in our trials.

3.4.3 Motion Energy-Based Pooling

Although the kernelized cut can derive segment boundaries that maximize intra-segment similarity and minimize inter-segment similarity, it has two drawbacks: (1) The kernelized cut is performed based on the similarity matrix, which is computationally expensive to obtain. (2) The two iterations, i.e. first running sliding window-based pooling and then running graph cut, do not eliminate the "chicken-and-egg" problem, since temporal boundary localization and feature aggregation are computed from the same information source.

To overcome these drawbacks, we propose the motion energy-based pooling, which is inspired by the observation that most human motions can be categorized into two meta-classes, i.e., *moving actions* and *still poses*, depending on the motion energy. Compared to the kernelized cut-based pooling, this motion energy-based method is more efficient, since it detects segment boundaries only based on the motion energy, which is fast computed from cluster IDs. Also, the motion energy can be considered as another information source, so that the "chicken-and-egg" problem tends to be avoided.

Like in [141], for a continuous human motion, we represent motion energy as a 1D function over time. Based on this 1D motion energy function, we determine temporal segments, and categorize these segments into the two meta-classes. Same with our sliding window-based pooling and the kernelized cut-based pooling methods, the input to this motion energy-based pooling is also a time sequence of encoded local features, as well as their individual cluster IDs derived from our dynamic clustering algorithm.

The 1D motion energy function is calculated based on the cluster IDs that the encoded input features belong to. We employ a moving window with size T_m and 1-frame stride to sequentially compute *the number of transitions between clusters* divided by the window size. Next, we detect peaks on the motion energy curve, retrieve windows around these peaks as the time periods of *moving actions*, and regard the remaining regions as *still poses*. We would like to emphasize that peaks in this 1D motion function are much easier and reliable to detect. Videos of daily livings usually contain long-term still poses, which cause the motion energy function staying at zero for a long time. These wide basin structures appear frequently, regarding them as valleys and determining their locations

are ambiguous and not reliable. Afterwards, in each individual segment, we fuse the encoded raw features (encoded body skeleton locations for example) by averaging, in order to represent the action in the segment. Empirically, setting T_m to the frame rate is without loss of generality and yields good results in our trials.

Here we give an example of how to compute the 1D motion energy function from the cluster ID sequence. We assume that the cluster sequence ID is $1, 1, 2, 2, 3, 2, 2, 4, 4, 4$, and the moving window size T_m is 3. With reflection padding on the two ends of the cluster ID sequence, i.e. appending another "1" to the left and another "4" to the right, the jumps in the cluster ID sequence is $0, 1, 1, 1, 2, 1, 1, 1, 0, 0$. Thus, the 1D motion energy is $0, \frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \frac{2}{3}, \frac{1}{3}, \frac{1}{3}, \frac{1}{3}, 0, 0$. Figure 3.4 illustrates the motion energy-based pooling. The raw input is a time sequence of body skeletons, each of which is represented by the concatenation of 3D joint locations. After dynamic clustering and feature encoding, the input to the motion energy-based pooling is the sequence of encoded body features, as well as their cluster IDs.

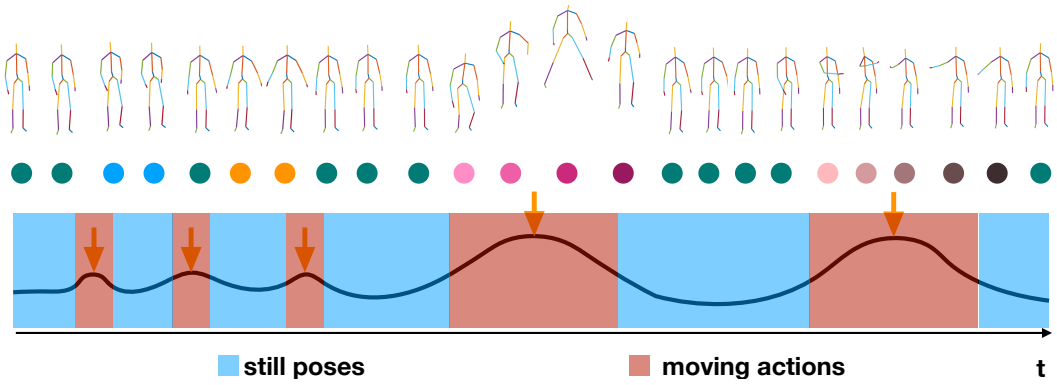

Figure 3.4: Illustration of the motion energy-based pooling. The color of dots denotes the cluster ID, the curve denotes the motion energy measure and the arrows denote the detected peaks. One can see that the moving actions and still poses are differentiated.

3.5 Temporal Segment Clustering

In the HDC pipeline (see Fig. 3.1), the last step is to cluster the obtained action patterns of temporal segments, which corresponds to classification in the conventional BoW

pipeline. After local temporal pooling, we employ another dynamic clustering module, which takes the high-level action patterns as input, to assign each individual temporal segment a cluster ID. One can note that we have 2 dynamic clustering modules in the HDC pipeline. The first one is used for codebook learning, and the second one is used here for temporal segment clustering.

The temporal segment clustering procedure can either be performed in a sequential manner, i.e. running dynamic clustering sequentially as taking the sequence of high-level action patterns as input, or be performed in a concurrent manner according to some meta information. For example, concurrent dynamic clustering can use the meta information from the motion energy-based pooling. One can recall that the motion energy-based pooling not only extracts high-level action patterns from temporal segments, but also categorizes these temporal segments into two meta classes. In this case, we run two separate dynamic clustering steps concurrently, in order to assign independent cluster IDs for segments of *moving actions* and segments of *still poses*, respectively. In the end, the cluster IDs of different meta actions are unified. For example, there are three segments of *moving actions* with cluster IDs “a1”, “a2”, and “a3”, and there are two segments of *still poses* with cluster IDs “b1” and “b2”. To unify the cluster IDs, we change “b1” and “b2” to “a4” and “a5”.

3.6 Experiments

Considering the applications for elderly people healthcare, in our study, we conduct two experiments on temporal action segmentation and novel action detection in omnidirectional videos, respectively. An effective solution to temporal action segmentation is highly useful for parsing and understanding composite daily activities. An effective solution to novel action detection in omnidirectional videos can be used for detecting abnormal behaviors, such as fainting and falling, from surveillance cameras in smart homes or hospitals.

3.6.1 Temporal Action Segmentation

Datasets. We use the **CMUMAD** [142] and the **TUMKitchen** [143] datasets to evaluate the performance of our methods for temporal action segmentation. **CMUMAD** contains 40 recordings (video and motion capture data) from 20 subjects, and each recording comprises 35 different actions and a null action (standing still) in between. **TUMKitchen** contains 20 recordings from multiple types of modalities such as videos, motion captures and on-off sensors on the door. The action labels are annotated for the torso and the arms separately.

Evaluation metric. On the **CMUMAD** dataset we evaluate the performance on action segment detection in terms of *precision/recall/f1-score*. Following the work [142], we also exclude the standing action (null class) when evaluating the method performance. Since all actions only occur once in each recording, from the beginning of the recording a true positive detection is defined as a segment having at least 50% overlaps with the ground truth segment and the associated cluster index has not been detected before.

On the **TUMKitchen** dataset we evaluate how effective our algorithm can locate the segment boundaries. We don't use this dataset for action segment detection, because the annotation of arm motions is based on hand-object interaction, leading to many cases that the same arm pose can have different semantic action labels. The detection metric is also in terms of *precision/recall/f1-score*, and a true positive detection is the boundary detected within ± 7 frames (approx. ± 0.25 second) of a ground truth boundary. In addition, we report the score about the left and the right arms jointly, due to their symmetric musculoskeletal natures.

Input features. Our algorithm is generic, and is able to process diverse types of features from different modalities. In our experiment, we use the following features: (1) The output of the *fc8* layer of the pre-trained two stream deep neural model VGG-16 [103, 144], which is denoted as VGG16. (2) The concatenation of all the 3D coordinates of the joints in each individual frame, which is denoted as JointLocation. (3) The relative 2D angles between each two adjacent 3D body parts, which is denoted as RelativeAngle.

(4) The quaternion representation of the 3D rotations (e.g., yaw, pitch and row) of the joints, which is denoted as Quaternions. The evaluation metric and the input features are identical to the one used in [38] to enable a fair and direct comparison with the state-of-the-art. One can note that for **TUMKitchen** we don't use VGG16, since such feature is extracted from video frames without separating human body parts, but the annotation is separate for torso and arms.

Comparison between clustering methods w.r.t. temporal segment clustering. We use the torso motions in **TUMKitchen** to compare k-means, spectral clustering and dynamic clustering, since the torso motion only has two types, i.e. *StandingStill* and *Walking*. For fair comparison, for all three methods we use dynamic clustering for codebook learning and sliding-window based pooling in the HDC pipeline (Fig. 3.1) As shown in the top plot of Figure 3.5, the proposed dynamic clustering method for temporal segment clustering outperforms the other two methods over all the three body pose features.

Comparison between temporal pooling schemes. To validate the effectiveness of the proposed local temporal pooling methods, we further compare with the conventional sliding window approach (SW) with an empirically optimal window size. As shown in the second plot of Fig. 3.5, the f1 scores from the sliding window scheme are consistently smaller than the f1 scores from the kernelized cut pooling scheme and the motion energy pooling scheme with various input features. This fact indicates that our proposed local temporal pooling methods are effective, and robustly outperform the baseline method.

Comparing with the state-of-the-art. We compare our methods with the following existing methods: TSC [33], ACA [31], EMS [32] and DC [38]. The results on the **CMUMAD** dataset are shown in Table 3.2. HDC$_{ME}$, HDC$_{KC-S}$ and HDC$_{KC-B}$ denote our proposed hierarchical dynamic clustering with the motion-energy based pooling, the kernelized sequential cut-based pooling and the kernelized batch cut-based pooling method, respectively. Note that, DC [38] compared in Table 3.2 utilizes the prior

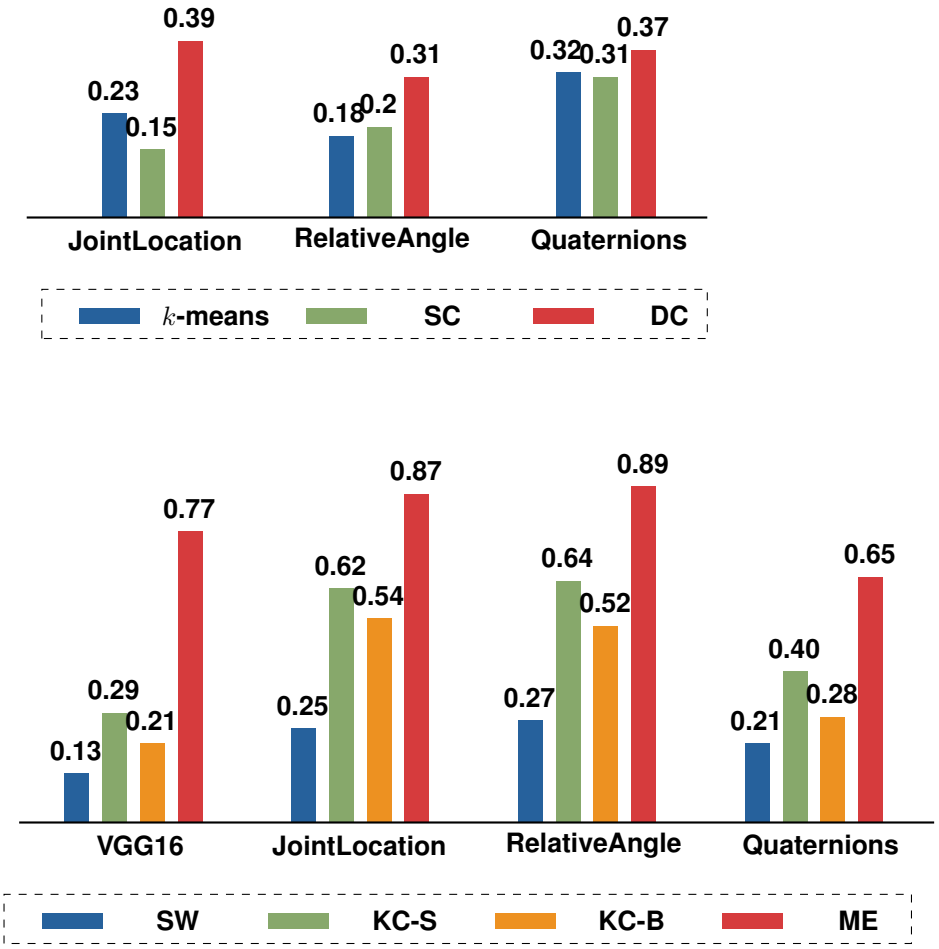

Figure 3.5: The f1-scores of different methods using features are shown here for comparison. One notes that the f1 score ranges from 0 to 1, and its value is the higher the better. **The top plot**: Comparison of clustering methods to cluster temporal segments of torso motions in **TUMKitchen**, where SC and DC are spectral clustering and dynamic clustering respectively. **The bottom plot**: Comparison of the temporal pooling schemes on **CMUMAD**.

knowledge to dedicatedly design the temporal pooling step for **CMUMAD**, whereas our proposed temporal pooling methods are generic and do not use any dataset specific information. Nevertheless, HDC$_{ME}$ outperforms other work for most of the input features considerably. A probable reason is that HDC$_{ME}$ can separate static body poses and moving actions, which fits the human motion action characteristics in **CMUMAD**. One can recall that the continuous human motion in **CMUMAD** has consecutive actions with the pose of standing still in-between.

Method	VGG16 [144]	JointLocation	RelativeAngle	Quaternions
TSC [33]	0.01/0.20/0.02	0.10/0.30/0.15	0.05/0.29/0.09	0.05/0.29/0.09
ACA [31]	0.56/0.66/0.61	0.55/0.68/0.61	0.51/0.65/0.57	0.55/**0.66**/0.60
EMS [32]	0.67/0.73/0.70	0.34/0.78/0.47	0.47/0.89/0.62	0.60/0.51/0.55
DC [38]	0.44/0.60/0.51	0.82/0.86/0.84	0.63/0.64/0.63	0.63/0.52/0.57
HDC_{KC-S}	0.23/0.41/0.29	0.50/0.82/0.62	0.52/0.82/0.64	0.31/0.58/0.40
HDC_{KC-B}	0.14/0.40/0.21	0.39/0.86/0.54	0.37/0.85/0.52	0.18/0.61/0.28
HDC_{ME}	**0.72**/**0.82**/**0.77**	**0.86**/**0.88**/**0.87**	**0.87**/**0.91**/**0.89**	**0.76**/0.57/**0.65**

Table 3.2: Comparison with the state-of-the-art on the **CMUMAD** dataset. The results are shown in the format of *precision/recall/f1-score*. Best are in boldface.

The results on the **TUMKitchen** dataset are shown in Table 3.3. For the **TUMKitchen** dataset, the input body pose features are derived from the 3D motion capture data rather than from video frames, and the 1D motion energy function is computed according to the method demonstrated in Sec. 3.4.3. Therefore, the 1D motion energy sequence has the same temporal length with the input time sequence. In terms of the *f1-score*, the proposed HDC_{KC-B} method shows superior performance to other methods. Interestingly, the performance of the HDC_{ME} method is not as outstanding as it is on the **CMUMAD** dataset. Our observation is that in the **TUMKitchen** dataset, the recorded persons perform fine-grained and long-time actions, and hence the variation for the 1D motion energy function frequently stays at a low value range. In this case, the motion energy curve does not provide strong signals on how to segment the continuous motion. Nevertheless, the performance of HDC_{ME} is still comparable to other state-of-the-art methods, especially for the JointLocation of torso.

3.6.2 Fainting Detection in Omnidirectional Videos

Computer vision systems have great potentials to be employed in smart home, hospital and other environments for healthcare purpose [145, 146]. Compared with wearable sensors like smart wristband, a camera network can perceive human bodies in an unobstructing manner and one can extract rich information of human behaviors from imagery data. To avoid installing a complicated camera network, in many applications an omnidirectional camera is sufficient, since its angle of view is 360-degree wide to perceive the entire indoor environment and from the captured videos one can effectively

Method	body part	JointLocation	RelativeAngle	Quaternions
TSC [33]	Torso	0.12/0.04/0.06	0.28/0.10/0.15	0.31/0.19/0.24
	Arms	0.42/0.28/0.34	0.29/0.38/0.33	0.28/0.56/0.37
ACA [31]	Torso	0.19/0.01/0.02	0.30/0.02/0.04	**0.40**/0.03/0.06
	Arms	0.36/0.08/0.13	0.36/0.10/0.16	0.34/0.09/0.14
EMS [32]	Torso	0.13/0.06/0.08	0.28/0.15/0.20	0.37/0.12/0.18
	Arms	0.26/0.12/0.16	**0.38**/0.27/0.32	0.34/0.09/0.14
DC [38]	Torso	**0.46**/0.15/0.21	**0.34**/0.12/0.18	**0.40**/0.26/0.32
	Arms	**0.49**/0.30/0.37	0.27/0.64/**0.38**	0.33/**0.68**/0.44
HDC$_{KC-S}$	Torso	0.24/**0.67**/0.35	0.23/0.44/0.30	0.27/0.49/0.35
	Arms	0.26/**0.52**/0.35	0.25/0.50/0.33	0.38/0.35/0.36
HDC$_{KC-B}$	Torso	0.32/0.54/0.40	0.23/**0.52**/**0.32**	0.29/0.64/**0.40**
	Arms	0.44/0.45/**0.44**	0.26/**0.72**/**0.38**	**0.44**/**0.46**/**0.45**
HDC$_{ME}$	Torso	0.42/0.54/**0.47**	0.24/0.45/0.31	0.23/0.39/0.29
	Arms	0.37/0.41/0.39	0.30/0.32/0.31	0.31/0.35/0.37

Table 3.3: Comparison with the state-of-the-art on the **TUMKitchen** dataset. The results are shown in the format of *precision/recall/f-score*. Best are in boldface.

analyze human behaviors despite the lens distortion [147]. Therefore, with only one omnidirectional camera, human hehavior analysis can be effectively performed.

The task of abnormal human behavior detection has been studied in several decades, which is highly essential for elderly people healthcare applications. According to a recent review [148], abnormal behaviors can be defined from the perspective of statistics. We can regard some behaviors abnormal, if they are relatively rare and occur less frequently than other behaviors. For example, within a bedroom, normal behaviors of an old man can include walking, sits on a sofa, sleeping on the bed, drinking water, and so on, since these behaviors occur frequently in daily living scenarios. On the other hand, lying on the ground is abnormal, since it occurs rarely in daily lives, and also can indicate a dangerous situation of that old man. From the pattern recognition perspective, once we have a large number of action patterns, abnormal behavior detection can be regarded as an anomaly detection problem [149]. Namely, we need to find action patterns that occur less frequently within a large population of action patterns. One can note that it is necessary to quantify “less frequently”, e.g. defining a threshold on the occurrence, so as to classify a behavior to be either normal or abnormal, such as in [49]. In many studies, the concepts of abnormal behavior detection and novel behavior detection are interchangeable. In our study, we define a novel behavior as a behavior that occurs *only*

once until the current observation time. Therefore, under our definition, a novel behavior is a specific kind of abnormal behaviors, and defining a novel behavior is temporally causal.

As discussed in the previous sections, the proposed HDC method can partition a continuous human movement into several disjoint temporal segments, and assign each temporal segment a cluster ID. When a cluster ID occurs only once until the current observation time, we regard that temporal segment contains a novel behavior. In this experiment, we focus on fainting detection. Normally, a person, who suffers from fainting after a series of actions, falls to the ground and retains the static "lying" pose for a long time. Based on such prior knowledge, we evaluate the performance of fainting detection by checking whether the obtained temporal segment in the end of the video has a novel cluster ID or not, as well as how well the temporal range of this segment matches the ground truth. In the following we present how to use our HDC method to detect fainting in omnidirectional videos.

Dataset. For fainting detection we use the first scenario of **BOMNI** [147], which consists of 5 people performing 6 actions resulting in total 10 videos. Each video only contains the behaviors of one person. The second scenario of **BOMNI** records interacting behaviors between three people, and hence is out of our current scope. The annotation of each video consists of the bounding box of the subject in each frame and 6 actions performed by the subject, which are sitting, walking, drinking, washing-hands, opening-closing-door and fainting. Figure 3.6 shows ten example frames.

Input features. First we use the pre-trained mask R-CNN model [150] to extract the mask of the subject in each individual frame, according to the annotated bounding box. We discard video frames when the annotated bounding box does not exist or the mask R-CNN fails to produce a mask.

Then we resize the detected masks to patches of 40x40 of pixels, and perform distance transform following [49]. The feature vector is derived via vectorizing the transformed mask. Fig. 3.7 illustrates extracting the mask feature.

Figure 3.6: Ten exemplar frames of the first video in the **BOMNI** dataset. In each frame, the annotated bounding box, the action label and the mask detected by Mask-RCNN [150] are presented.

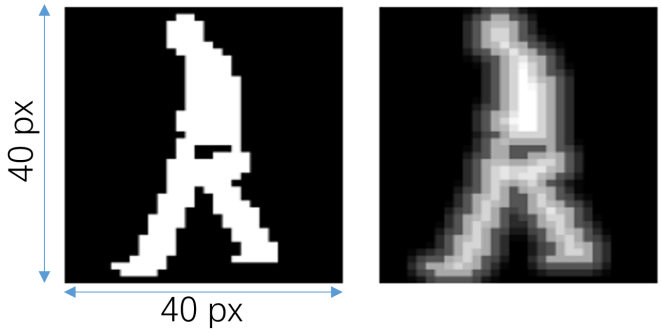

Figure 3.7: Mask features of the person, proposed by [49]. From left to right: (1) The person mask and (2) the distance transform result.

Evaluation metric. We use the accuracy to evaluate the detection quality at the frame level, which is computed as the true positives divided by the number of frames belonging to the *fainting* action in the ground truth. The true positives are defined as the frames, which both belong to "novel" actions and the associated label (cluster index) is the majority in the parsing result within the period of *fainting*. In our setting, a "novel" action is a segment with a label that first appears along the time dimension. For example, if we have a sequence of 6 frames with action labels (or cluster indexes) $\{a, a, b, a, a, c\}$, the frames are annotated as $\{1, 0, 1, 0, 0, 1\}$ for labeling novel behaviors. Therefore, such evaluation metric is sensitive to temporal orders of action occurrences.

Results. The results of our methods as well as other state-of-the-art parsing approaches are shown in Table 3.4. One can see that HDC$_{ME}$ is comparable with ACA and superior to others. However, HDC$_{ME}$ runs significantly faster than ACA. Since the motion-based pooling mainly relies on processing the cluster indices, its computational load is considerably smaller than other pooling methods which require to compute the data similarity matrix. Also, comparing with dynamic programming used in ACA, subspace clustering used in TSC and k-means used in DC, the dynamic clustering modules in HDC lead to faster computation.

Algorithm	TSC [33]	ACA [151]	DC [38]	HDC$_{KC-S}$	HDC$_{KC-B}$	HDC$_{ME}$
Accuracy	0.28	**0.99**	0.33	0.81	0.81	0.92
Runtime (second)	1.63	2.69	0.16	0.18	0.31	**0.07**

Table 3.4: The results on the **BOMNI** dataset. The best ones are in boldface.

3.7 Discussion on Hierarchical Dynamic Clustering

In this chapter, we propose the hierarchical dynamic clustering (HDC) method to parse human actions in an unsupervised manner. Our HDC method is inspired by the *bag-of-visual-words* pipeline, but incorporates new modules which are more suitable to process multi-dimensional time series data. Specifically, we design the dynamic clustering algorithm for codebook learning and temporal segment clustering, and propose local temporal pooling schemes to detect segment boundaries and extract segment patterns. Our experiments show that the proposed HDC method can parse continuous human actions effectively, and outperforms other state-of-the-art unsupervised methods, as indicated by higher metric scores and computational speed.

In the entire HDC pipeline, there exist a number of hyper-parameters to determine. Specifically, in the dynamic clustering algorithm, one needs to specify the threshold value δ_r (or c). In the feature encoding step, one needs to specify the value of λ in the soft-assignment scheme. In the kernelized cut-based pooling scheme, one needs to set the time window size to compute the similarity matrix. In the motion energy-based pooling scheme, one needs to specify the time window size to calculate times of

cluster transitions. Empirically, one can specify the values of these hyper-parameters according to the characteristics of each module in HDC, which considerably reduce the difficulties of using HDC in practice. For example, in dynamic clustering the threshold hyper-parameter δ_r can be determined as half of the minmum distance between two clusters in the validation dataset. Also, in our trials, fixing the time window size equal to the video frame rate in local temporal pooling can produce good performance in general.

In the HDC pipeline, we propose the kernelized cut-based method and the motion energy-based pooling method for local temporal pooling. The former method finds optimal cuts on the similarity matrix, and hence is generic for various human motions. However, due to the similarity matrix computation, the kernelized cut-based pooling is less efficient than the motion energy-based pooling method, which detects segment boundaries only based on cluster indexes. In addition, the motion energy-based pooling is highly effective to separate static poses and moving actions, and hence outperforms other methods in terms of action segmentation in **CMUMAD**, fainting detection in **BOMNI** and torso movement segmentation in **TUMKitchen**, which comprise recordings with "standing still" and "moving actions". Since this case is very common in daily living scenarios, we expect that the proposed motion-energy pooling method has large potentials in practice.

In summary, we have proposed hierarchical dynamic clustering for unsupervised human action parsing in untrimmed videos, and employed this HDC method in the tasks of temporal action segmentation and fainting detection. The experimental results verify the effectiveness and efficiency of HDC, and reveal its potentials for applications of vision-based elderly people healthcare. Nevertheless, some issues still remain. For example, our HDC method is fully bottom-up, and lacks top-down information to correct errors at the bottom level. Therefore, if a bottom module does not perform well, e.g. the codebook is not well learned, its inferior performance will influence the top level and can never be corrected. Also, the raw input features are handcrafted, which might not fit well with the nature of input data. These drawbacks encourage us to propose solutions based on deep learning methods, which enables back-propagation to provide top-down information to improve bottom-level modules, and derive data-adaptive features from the input. Therefore, in the next chapter, we will investigate how to use deep learning method to realize human action parsing in untrimmed videos.

4Supervised Human Action Parsing via Deep Neural Networks

In Chapter 3, we have presented our hierarchical dynamic clustering (HDC) method for unsupervised human action parsing. Despite its effectiveness for segmentation and abnormal behavior detection, the result does not have semantic meanings. For example, the proposed HDC method detects abnormal behaviors only based on the occurrence frequency, and hence cannot understand what exactly the detected abnormal behaviors are. This drawback can cause severe problems in practice. For example, “running in the bedroom” and “fainting on the ground” can be both detected as abnormal behaviors by HDC due to their rare occurrences, and are simply regarded as Cluster A and Cluster B, respectively. However, treating them equally is definitely not reasonable in practice, because “fainting on the ground” is much more dangerous. To overcome such drawback, we aim at proposing an automatic algorithm, which can not only segment action temporally, but also understand actions (i.e., assigning semantic labels) in individual temporal segments. As a result, “running in the bedroom” and “fainting on the ground” can be treated differently. The system can remind the person to keep caution on slippery areas in case of “running in the bedroom”, and rises emergency alarms to inform healthcare people in case of “fainting on the ground”. Therefore, in this chapter, we address human action parsing in a supervised manner: Given an untrimmed video of human actions, we aim at assigning each individual frame an action label. With such supervised action parsing method, we can perform video trimming, action recognition, action detection and other tasks under an unified framework.

As discussed in Chapter 1, supervised human action parsing can be performed in a two-stage approach. In the two-stage approach, our proposed HDC method can be first applied to produce action patterns in individual temporal segments, and then an off-the-shelf action recognition method can be used to assign each temporal segment an action label. Although such two-stage approach can somehow overcome the drawbacks of unsupervised human action parsing stated above, it is still sub-optimal: (1) The

semantic information (e.g. the annotated action labels) only influences the second stage. The segmentation process in the first stage ignores semantic meanings, and has high risk of producing segments which are not consistent with the current semantic scenario. (2) The features are hand-crafted, which may lack representativeness for the current semantic scenario. To overcome drawbacks of the two-stage approach, in this chapter we investigate an end-to-end one-stage approach for action parsing based on deep neural networks, so that the annotated semantic labels can supervise learning in all stages via back-propagation, and influence low-level feature extraction in a top-down manner. Once a deep neural network is trained, tuning hyper-parameters by hand during the inference phase is usually trivial.

This chapter is organized as follows: First, we introduce a spatiotemporal convolutional encoder-decoder network, with the input of an image sequence (i.e. a video) and the output of frame-wise action labels. Rather than training the spatial network and the temporal encoder-decoder separately as in [56, 59], we train the entire network end-to-end and use it to recognize falls in videos. Second, to extract high-order information from features for parsing fine-grained actions, which are very common in daily living scenarios and important for our healthcare purpose, we propose local temporal bilinear pooling methods from two perspectives. When employing our bilinear pooling method as layers in a large-scale deep neural architecture, we achieve state-of-the-art performance on various fine-grained action parsing benchmarks.

4.1 Spatiotemporal Convolutional Encoder-Decoder Network

It is reported in the literature, e.g. in [59] and [60], that spatiotemporal convolutional encoder-decoder networks are highly effective for semantic temporal action segmentation, and outperform other network architectures such as deformable convolutions [152] and RNNs [153]. Therefore, we investigate such network architecture in our study.

A typical spatiotemporal convolutional network comprises two parts, i.e. a spatial convolutional network and a temporal convolutional encoder-decoder network, which are illustrated in Figure 4.1. Given an image sequence as input, the spatial network converts

each individual frame (or a short video snippet) into a feature vector, so that the image sequence becomes to a feature vector time sequence. Then, a temporal convolutional encoder-decoder network is applied to compute temporal correlations of these feature vectors, and produces a time sequence of action labels.

Several studies like [57], [59] and [60] train the spatial network and the temporal network separately. An important advantage of such separate training is that one can pre-train a complex spatial network to transfer the knowledge from large-scale datasets, such as ImageNet [154] or Kinetics [11], to the small target datasets for generaliability [155]. Although such pre-training approach is effective in most cases, such knowledge transfer process may fail, especially when a large domain gap between the target datasets and the publicly available large-scale datasets. For example, when we aim at recognizing falls from videos captured in certain indoor environments, a complex spatial network pre-trained on Kinetics may produce unrepresentative features, since Kinetics contains few samples captured in similar environments with our target environment. Therefore, in our study, we not only consider complex spatial networks which are pre-trained on these large-scale datasets, but also design relatively small spatial network architectures, according to the natures of the target datasets, so as to obtain representative spatial features for the target scenarios.

4.1.1 Spatial Convolutional Network

In principle, the spatial network can be any kind of deep neural network for image classification like VGG networks [58], residual networks [156], Alexnet [157], Inception networks [158, 159, 160], or for video classification like I3D [11] and ST-Resnet [161]. With a network pre-trained on large-scale datasets (e.g. Imagenet [154] or Kinetics [11]), one can convert a video frame or a very short video snippet (e.g. 10 frames) into a 1D feature vector, as output of a fully connected layer or vectorizing the output from an intermediate convolutional layer.

Training these large-scale deep neural networks usually requires large-scale datasets; otherwise, over-fitting is likely to occur. When the number of training samples for a specific task is not sufficient, a conventional approach is to fine-tune a neural network

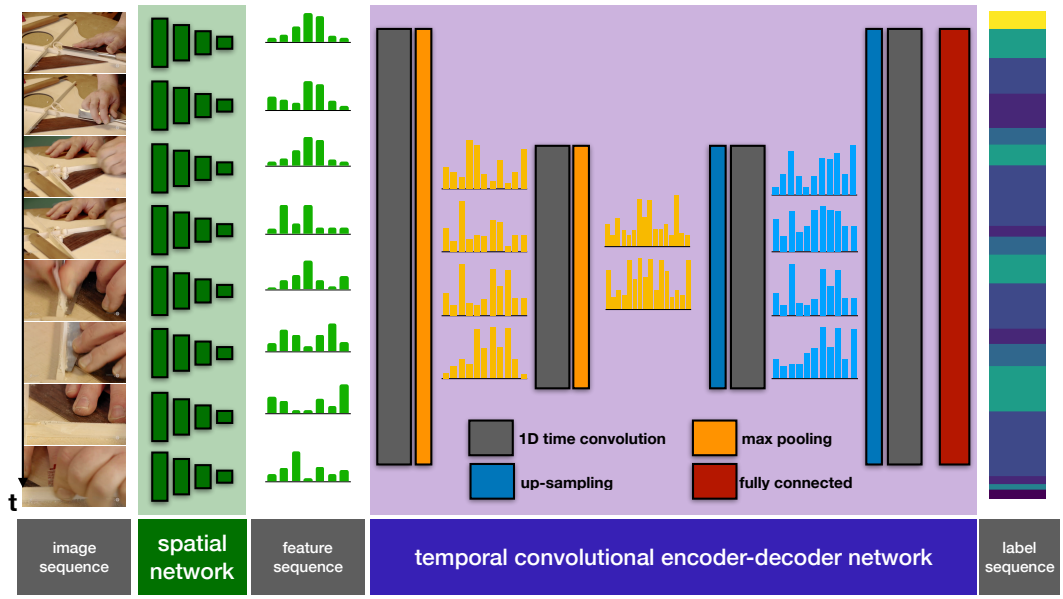

Figure 4.1: Illustration of a spatiotemporal encoder-decoder network. Given an image sequence, a spatial network is applied to each individual frame to extract frame-wise feature vectors. Then, the temporal encoder-decoder network computes local temporal correlations, and produces a time sequence of action labels. Each action label is assigned to each individual frame of the input video.

which is pre-trained on a large-scale dataset such as ImageNet. For example, in [58], the original VGG network for image classification is fine-tuned to regress 2D bounding boxes for object detection in images. When performing back-propagation, only the weights in layers for 2D bounding box regression are updated. Therefore, the knowledge is transferred from image recognition to object detection.

Nevertheless, the functionality of fine-tuning is still not clear. It is reported in a recent study [162], that training a deep neural network from random initialization (from scratch) can achieve a comparable performance with fine-tuning from ImageNet, with respect to object detection and instance segmentation on the COCO dataset. In [57], the spatial network has a similar architecture of the VGG16 network but with fewer layers, and is trained from scratch.

Therefore, when the scale of the deep neural network is much larger than the dataset, we use pre-trained networks to avoid overfitting. Otherwise, we train the network from

scratch, in order to learn consistent knowledge to our target scenarios. Figure 4.2 shows an instance of a spatial network architecture, which is designed by us to detect falls in videos [163].

4.1.2 Temporal Convolutional Network

Given a time sequence of 1D feature vectors, the temporal network computes temporal correlations of feature vectors, and produce a time sequence of action labels, each of which is assigned to each individual video frame. Conventionally, such sequence-to-sequence problem is solved by recurrent neural networks (RNNs), e.g. bi-directional LSTM [164]. However, in recent studies, it is reported that temporal convolutional networks with an encoder-decoder architecture can consistently produce superior results for human action analysis [59, 165, 60]. Therefore, we focus on investigating such kind of temporal convolutional networks to capture temporal structures in the data.

One can see an example of temporal encoder-decoder network in Figure 4.1. A typical temporal encoder-decoder architecture has three components, i.e. a temporal encoder, a temporal decoder, and a fully connected layer. The temporal encoder has successive "1D convolution $\rightarrow$ 1D pooling" structures. The 1D convolution is performed along the temporal dimension, and can be formulated by

$$y_t^d = \sigma\left(\left\langle \boldsymbol{w}^d, \oplus_{\tau\in\mathcal{N}(t)}\boldsymbol{x}_\tau \right\rangle + \boldsymbol{b}\right), \tag{4.1}$$

in which $\boldsymbol{x}_\tau$ denotes the input feature vector to the 1D convolution at time τ, $\oplus$ denotes feature concatenation, $\mathcal{N}(t)$ denotes the receptive field centered at time t, $\boldsymbol{w}^d$ denotes the d-th 1D convolution filter, $\langle\cdot\rangle$ denotes the inner product operation, $\boldsymbol{b}$ denotes the bias, $\sigma(\cdot)$ denotes a nonlinear activation function, such as ReLU [157] and leaky ReLU [166], and y_t^d denotes the d-th element of the output feature vector $\boldsymbol{y}_t$ at time t. The dimensionality of the output feature vector is determined by the number of filters in this 1D convolutional layer. One notes that this 1D temporal convolution is *non-causal*. As reported in [59], the non-causal convolution outperforms a causal version w.r.t. temporal action segmentation.

In addition, the 1D pooling layer can be demonstrated by

$$\boldsymbol{y}_t = \mathcal{L}_{\tau \in \mathcal{N}(t)} \boldsymbol{x}_\tau, \tag{4.2}$$

in which $\mathcal{N}(t)$ is the receptive field of the pooling operation centered at time t, and $\mathcal{L}$ is an algebraic operation such as element-wise maximum and average, corresponding to max pooling and average pooling, respectively. Without explicit mentioning, in our work, we use max pooling with the receptive field equal to 3 and the stride equal to 2, which halves the temporal resolution.

The architecture of the decoder is symmetric with the encoder, which has successive layers of "1D upsampling $\rightarrow$ 1D convolution". The upsampling operation can be regarded as an inverse operation with the temporal 1D pooling. Without explicit mentioning, in our work, the upsampling operation is performed by nearest-neighbour interpolation to double the temporal resolution.

After the last convolutional layer in the temporal decoder, a time-distributed fully connected layer with softmax output is applied to convert each feature vector to an action probability. Such fully connected layer is applied on each individual frame across time with shared parameters. Specifically, the fully connected layer can be formulated as

$$\boldsymbol{y}_t = softmax(\boldsymbol{W}\boldsymbol{x}_t + \boldsymbol{b}), \tag{4.3}$$

in which $\boldsymbol{x}_t$ is the t-th frame of the output from the temporal decoder, $\boldsymbol{W}$ and $\boldsymbol{b}$ are the parameters of the time-distributed fully connected layer, and $\boldsymbol{y}_t$ is the resultant action probabilistic distribution, the dimensionality of which is equal to the number of action classes in the training data. The final label is the action with the maximal probability.

An instance of the temporal encoder-decoder network is shown in Figure 4.2. One can see that the temporal encoder has two "1D convolution $\rightarrow$ 1D pooling" modules. The first convolution layer has 96 filters, and hence its output feature vector is 96-dimensional. The second convolution layer has 128 filters. The decoder has a symmetric architecture with the encoder. The output from the temporal decoder has the same feature dimension and the same temporal resolution with the input to the temporal encoder.

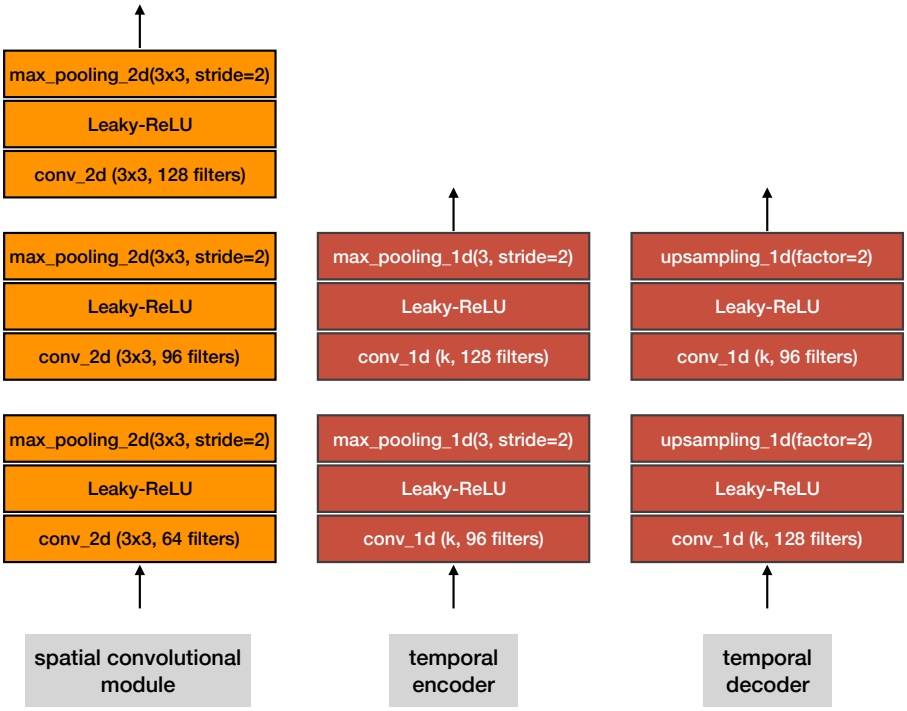

Figure 4.2: An architecture instance of the spatiotemporal convolutional encoder-decoder net. From left to right: An instance of the spatial network, an instance of the temporal encoder and an instance of the temporal decoder. In the temporal encoder and the temporal decoder, k denotes the size of the 1D convolution receptive field, i.e. the size of $\mathcal{N}$ in Eq. (4.1). We use different values of k in following sections, e.g. Sec. 4.2.2.

4.2 Experiment: Detecting Unintended Falls from Videos

Due to cognitive impairment or deficiencies of motor functionalities, unintended falls occur frequently in the group of elderly people, and can lead to severe or even fatal injuries [167, 168]. Therefore, to build up fall detection systems for elderly people healthcare, it is essential to recognize falls in an automatic and effective manner.

Fall recognition has been intensively studied in the past. If the human body dynamics has been precisely measured, identifying an unintended fall is straightforward. For example, one can recognize falls via measuring the vertical velocity of the human body towards the

ground. If the velocity is above a threshold, then a fall occurs. Consequently, researchers tend to propose novel solutions to capture the body configurations and motions. For example, the work of [169] uses a wearable tri-axial accelerometer to measure the body motion, and recognizes falls via a one-class support vector machine. The work of [170] develops a wearable system (mainly based on the accelerometer and GPS) to detect and localize falls in the wild. Wearable sensors enable measuring physical attributions of the human body in a precise and real-time manner. However, the sensors have to be physically attached to people, causing obstructive interventions to their daily living activities.

Computer vision technologies enable non-obstructive measurement of human body motions, and conduct behavior recognition only based on imagery data. The effectiveness is highly improved when deep convolutional networks trained on large-scale image datasets are employed. To recognize a fall, two families of methods can be considered: The first family attempts to capture precise body configurations over time, such as [171] and [172] for 2D pose estimation and [173] for 3D pose estimation. These pose estimation methods can replace the functionality of wearable sensors, but perform human body measurement in a non-contacting fashion. The second family, which is usually based on deep convolutional networks, aims at inferring the semantic content of the input data, via creating a mapping from the input data to the action labels end-to-end. For example, [103] yields an action label for an input sequence, [174] yields both action labels and temporal durations, and [59] produces frame-wise labels for temporal action segmentation. In this thesis, we focus on the second family of methods, since the end-to-end inference behavior does not need any intermediate step, e.g., training a classifier based on the captured body poses. In addition, the data annotation procedure only requires to assign action labels to frames/videos, which is much easier to implement in practice than annotating the key joints on the human bodies in each frame required by the first family of methods.

Although several relevant methods like [109] have been proposed, the underlying reasons of the effectiveness are still not clear. Therefore, beyond proposing a method for fall recognition, we aim at attaining insights of how the deep convolutional network recognizes falls. Our investigation is based on a family of convolutional encoder-decoder

networks, various types of input modalities such as RGB images, optical flows and 2D body poses, and recordings from different environments. One can see details of our investigation in Sec. 4.2.2.

According to our investigations, we discover: (1) Human body motion represented by the optical flow is highly informative for the network to recognize falls. (2) The network tends to learn human body-centered context, namely the appearance surrounding the human body, if the training samples have RGB frames. However, the network cannot get rid of the influence of the background context irrelevant to falling, and lacks generalizability across different environments. (3) The human-centered context and human body motion are complementary. (4) Inaccurate body pose information can degrade the performances.

4.2.1 The Convolutional Encoder-Decoder Network (CED)

We formulate fall detection as a binary classification problem, and expect to obtain frame-wise semantic labels, so that recognition and temporal localization can be solved simultaneously. Therefore, we use the convolutional encoder-decoder network, as demonstrated in Sec. 4.1.2. Compared to other studies, e.g. [59], we train the spatial network and the temporal network jointly, so that the learned features in intermediate layers are representative in our tasks of fall understanding.

The CED architecture has several advantages besides generating frame-wise labels: (1) The CED network can capture long-range temporal dependencies, and outperforms recurrent nets, e.g. bidirectional LSTMs [175, 55], w.r.t. temporal action segmentation [59] and motion prediction [165]. (2) The CED network can generate piece-wise constant label sequences directly, without post-processing steps like median filtering. (3) Comparing with recurrent neural nets, in our trials we find that CED is much easier to train and converges much faster. (4) Once CED is trained, the model can process sequences of arbitrary lengths.

In the following, we present details of our employed CED architecture for fall understanding in videos, which can be regarded as a special instance of the spatiotemporal

encoder-decoder network demonstrated in Sec. 4.1. The network architecture is illustrated in Figure 4.2.

The spatial convolutional module. This convolutional module aims at extracting the feature vector of each individual frame in the video. It consists of three convolutional blocks, and each block contains a 2D convolutional layer with an activation function, and a 2D max-pooling layer, following the architecture of the VGG networks [58, 59]. After each block, the spatial resolution is downsampled by the factor of 2. At the end of the module, the output 3D tensor is flattened to a 1D vector. The number of convolution filters are suggested by [57]. In our work, we use the leaky-ReLU [176] activation function, due to the superior performance to the standard ReLU function [166]. Moreover, the spatial convolution module is applied on each individual frame of the input tensor sequence, and has shared parameters across all frames.

The temporal convolutional module. Such temporal convolutional module takes the output from the spatial convolutional module as input, which is a time sequence of 1D feature vectors. This 1D feature vector sequence has the shape of [time, dimension], or a 3D tensor with the shape [batch, time, dimension], when considering mini-batch input.

The temporal encoder and the decoder has symmetric structures. The encoder comprises two blocks, each of which has a 1D temporal convolutional layer with the leaky ReLU activation function, and a 1D max pooling layer. According to our demonstration of temporal convolutional network in Sec. 4.1.2, the 1D temporal convolution computes temporal relations of frames in a receptive field $\mathcal{N}$ with the size k (specified in Sec. 4.2.2), and produces a time sequence of feature vectors with the feature dimension equal to the number of filters. The 1D max pooling fuses every three consecutive features with a two-frame stride along the time dimension, and hence halves the temporal resolution.

The fully connected module. The fully connected module consists of a fully connected layer, a dropout layer and a softmax layer, and is applied on individual frames in the output of temporal decoder with shared parameters. Due to our binary classification

setting, the output dimension of the fully connected layer can be 1 or 2. Here we use the two-dimensional output, since we expect that the insights derived from our work can be extended to multi-class classification problems straightforwardly. The dropout layer (with a keep ratio of 0.5) is used to avoid overfitting, and the softmax layer converts the scores to probabilities.

Network training. In our work, all the modules are trained jointly from scratch, in contrast to [59] that only trains the temporal encoder-decoder using the outputs from a pre-trained spatial network. For each frame, we compute the *cross-entropy* between the one-hot encoded ground truth label and the softmax output. Then the loss of the sequence is the sum of the cross-entropy values of all frames. After specifying the loss, the model parameters are learned via the Adam algorithm [177]. One reason of using Adam is its superior performance than the stochastic gradient descent method in many tasks, and that empirically Adam trains network faster. Implementation details are explained in Sec. 4.2.2.

Network explanation. To interpret how the deep neural network recognizes falls from videos, we employ attribution maps to visualize the importance of pixels in the input frames, given a specific network output. We use the following attribution maps: (1) DeepLIFT [115], which is computed by backpropagating the contributions of all neurons in the network to every input feature. (2) Integrated gradients [112], which is computed by fusing the gradients along small straightline path from the baseline (a black image in our case) to our input image. (3) saliency map [113], which is the magnitude of gradients w.r.t. to input pixels. The study of [112] summarizes the computations of these attribution maps, and develops the DeepExplain Python package.

4.2.2 Empirical Studies

To investigate how the deep convolutional network CED recognize falls, we propose 4 tasks, and for each task the quantitative results are shown by cross-validated frame-wise accuracies and the qualitative results are shown by attribution maps.

Figure 4.3: Illustration of falls in videos of the **Le2i** dataset. From left to right: (1) Example frames of falls in *home* and *coffee room*. (2) The statistics of time durations of falls across all videos, in which the x-axis denotes the fall duration, the y-axis and the bins show the normalized occurrence frequencies and the curve shows the fitted distribution. One can see that the maximal duration of falls is smaller than 45 seconds.

Datasets. We use the **Le2i** Fall detection dataset presented in [178], which is built using a single camera in realistic surveillance setting containing static background, illumination variations, occlusions by furniture, different appearances of the subjects, different types of falls (e.g. falling forward, falling while sitting, etc.), and other factors that simulate falls in daily lives. The video has spatial resolution of 320$\times$240 of pixels and is captured with 25 fps. Each video is annotated in a frame-wise fashion.

The original dataset contains 4 environments, i.e. *home*, *lecture room*, *coffee room* and *office*. We only use the recordings from *home* and *coffee room* in our study, since recordings from other environments do not have annotations. For each of the two environments, there exist two groups of recordings.

In order to balance the number of fall and not-fall frames, we extract a video snippet consisting of frames before, during and after the fall from each video containing falling. Video trimming is based on the statistics of time durations of falls, which is shown in Figure 4.3. Specifically, the extracted snippet has 60 frames (2.4 seconds), starting from $T - 49$ to $T + 10$, where T is the timestamp of the last frame of fall in the video. Since the maximal duration of falls in **Le2i** is 45 second, the extracted 60-frame snippets cover the entire fall durations.

	Our processed **Le2i** dataset [178]			
high-level split	*home*		*coffee room*	
low-level split	group 1.1	group 1.2	group 2.1	group 2.2

Table 4.1: Organization of our processed **Le2i** dataset.

Depending on the recording environment, we perform a *high-level* splitting to divide the dataset into 2 folds, each of which contains recordings from either *home* or *coffee room*. Since there are two groups for each environment in **Le2i**, we perform a *low-level* splitting to divide the dataset into 4 folds, i.e. group 1.1, group 1.2 in the *home* environment, and group 2.1 and group 2.2 in the *coffee room* environment. Therefore, the *high-level* splitting can be used for cross-environment validation, and the *low-level* splitting can be used for cross-validation under small environment variations. As a result, we obtain a new dataset incorporating 99 video snippets with 2 *high-level* splits and 4 *low-level* splits, which are listed in Table 4.1.

Input modalities to the network. Besides the RGB frames, we also compute time differences, TV-L1 optical flows [179], and heat maps of human body joints[1] [180] [172], which are regressed from RGB images. For fast computation, we downsample the spatial resolution of video frames to 56×56 pixels.

Similar to [59], the network input contains a stack of frames from the original data sequence. Denoting the RGB image sequence as $\{I_t\}$, the optical flow sequence as $\{w_t\}$ (values within $[-20, 20]$) and the sequence of heat maps as $\{s_t\}$ (values within $[0, 1]$), the modalities used in our experiments are shown in Table 4.2.

The **RGB+TimeDifference** modality is suggested by [59], in which the RGB frames encode the appearances of the visual scene and the time differences have the functionality of attention. Image standardization (a.k.a. z-normalization) is performed frame-wisely, in order to eliminate the influence of illumination changes. Since the background is static, **TimeDifference** and **Optical Flow** focus on the human body, while **TimeDifference** does not measure directional human body motions. In addition, the pose information

[1] The MPII body model has 14 keypoints and hence the method generates 14 pose heat maps for each image. In our experiment, we average these 14 heat maps to one map.

Table 4.2: The input modalities used in our experiments, in which the **Pose+Optical Flow** modality uses the normalized optical flow $\tilde{w}_t$, i.e. $\tilde{w}_t = w_t/20$ (see text).

Modalities	Format of each frame
RGB+TimeDifference	$\{I_{t-1}, I_t, I_{t+1}, I_t - I_{t-1}, I_{t+1} - I_t\}$
TimeDifference	$\{I_t - I_{t-1}, I_{t+1} - I_t\}$
Optical Flow	$\{w_{t-1}, w_t, w_{t+1}\}$
Pose	$\{s_{t-1}, s_t, s_{t+1}\}$
Pose+Optical Flow	$\{s_{t-1}, \tilde{w}_{t-1}, s_t, \tilde{w}_t, s_{t+1}, \tilde{w}_{t+1}\}$

is represented by the pose heat map produced by the pre-trained network proposed in [180, 172], in which the heat map of a body joint is the probability of the joint locations within the image domain. It is reported that the pose heat map is beneficial for person re-identification and tracking [181]. When combining optical flow and pose, we expect that the pose heat map works as an attention mechanism, encouraging the network to learn motion features around the body key points.

One can note that the **Pose+Optical Flow** modality uses the normalized optical flow $\tilde{w}_t$, which is computed by $w_t/20$ and hence ranges within $[-1, 1]$. In this case, the ranges of the pose heat map and the optical flow are similar. We find that such flow normalization process is beneficial in our trials. A probable reason is that the normalization leads to similar ranges of convolution parameters for the flow and the pose map in **Pose+Optical Flow**.

Implementation. The implementation is based on TensorFlow. The batch size is fixed to 8, meaning 8 tensor sequences are fed to the network for one iteration. The Adam algorithm is used to train the model [177], in which the initial learning rate is 0.001 and other parameters are set to the TensorFlow default values, which are $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The learning rate is decayed every 10 epochs by $0.001 \times 0.9^{\lfloor \frac{epoch}{10} \rfloor}$, and training terminates after 100 epochs. In our trials, more iterations lead to comparable or worse results.

In addition, attribution maps for network explanation are computed based on the DeepExplain library [112], which is implemented in Tensorflow.

Evaluation. Rather than performing model selection as in [59], we use a family of network instances to verify whether some conditions can consistently influence the performances. We vary two influential factors in the network architecture, i.e., the non-causal temporal convolution receptive field size k, and the temporal length of the input sequence l determining the upper limit range of the temporal structure that the network can learn. In our experiments, we use the network instances with $(k, l) \in \{(3, 8), (3, 16), (3, 32), (5, 16), (5, 32), (7, 16), (7, 32)\}$.

For the *high-level* splitting, 2-fold cross-validation is performed, in which each network instance is trained on the first fold and validated on the second, and vice versa. Then, for each network instance, the two validated accuracies are averaged to derive the cross-validated accuracy. For the *low-level* splitting, 4-fold cross-validation is performed in an identical manner. Since each network instance associates with an accuracy value, the quantitative performance of the CED model is presented in terms of a box plot.

The qualitative results are shown by attribution maps, i.e. DeepLIFT [115], integrated gradients [112] and saliency maps [113]. For visualization purposes, each attribution map is superimposed to the input image edges, which are obtained by Canny detector [182].

Task 1: Investigating the cross-environment generalization. In this task, we aim at investigating the generalization of recognition performance across environments, namely, how the CED performs, if training samples and testing samples are collected from different environments. Therefore, we conduct a 2-fold cross-validation procedure based on the *high-level* splitting, and use **RGB+TimeDifference**, **TimeDifference** and **Optical Flow** as the input modalities. The results are shown in Figure 4.4.

Box plots summarize the resulting data. One can see that **RGB+TimeDifference** performs inferior to **TimeDifference** and **Optical Flow**, and **Optical Flow** outperforms **TimeDifference**. In addition, the attribution maps from four testing recordings consistently show that many pixels on the background can strongly affect the network inference process.

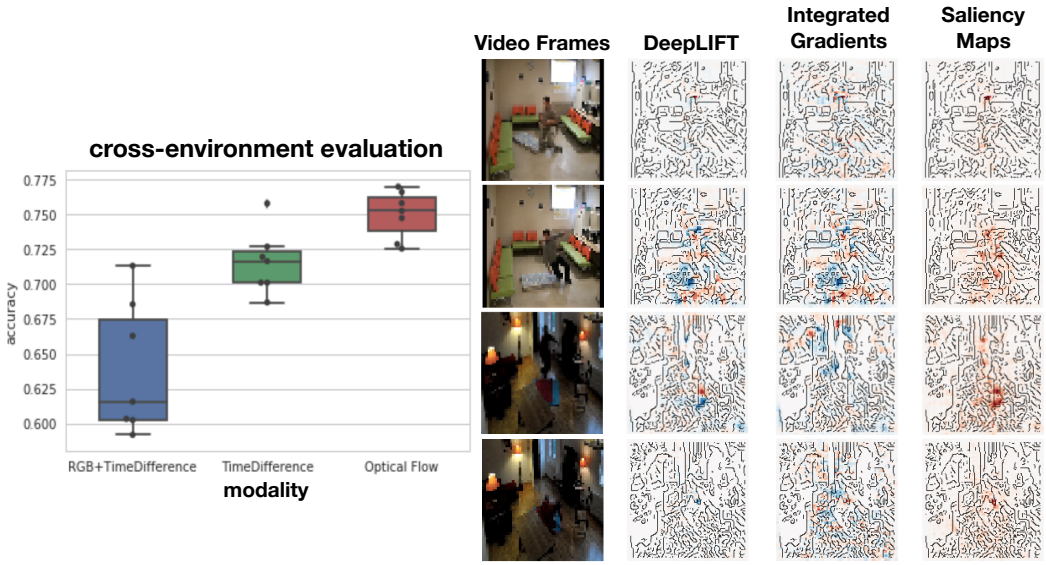

Figure 4.4: Experimental results of Task 1. From left to right: (1) Quantitative results of the 2-fold cross-validation, where the results from each network instance are shown as black dots in parallel to the box plots. In each box plot, the bar inside the box denotes the median, and the box shows the interquartile range (IQR) and the samples between whiskers with $1.5\times$IQR are inliers. (2) The attribution maps of frames from four test cases in the dataset are shown, where the red color and the blue color denote positive and negative contributions on the probability of falling, respectively. Image edges from the Canny detector are presented as well for visualization purposes.

The network with **RGB+TimeDifference** performs just slightly better than random guess, under the binary classification setting. The attribution maps show that irrelevant static background information has strong influence on fall recognition, and hence we consider that the network does not discard irrelevant background information automatically during training, and leads to degraded generalization across environments. On the other hand, **TimeDifference** and **Optical Flow** can discard static background information from the input video, and encourage the network to learn human body-centered features,

especially the motion features. Their superior performance can indicate that body motion contains more representative information of falls.

Task 2: Investigating the influence of training samples. Here we aim at investigating the benefits of training samples collected from the testing environment. In addition to the 2-fold cross-validation as presented in Task 1, we perform 4-fold cross-validation based on the *low-level* dataset splitting (see Table 4.1), and hence training samples have recordings from the testing environment. The employed input modality is **RGB+TimeDifference**, and the results are shown in Figure 4.5. The reason of only using **RGB+TimeDifference** is that other modalities used in Task 1, namely, **TimeDifference** and **Optical Flow**, are environment-independent and cannot reveal the influence of changing environments.

These box plots show that the using training videos captured from the testing environments can largely improve the performances. Indicated by all attribution maps on the right, we can find that the influential pixels become more human body-centered. Noticeably, one can see that the influential pixels tend to locate around the human body contours, rather than directly on the human body. This fact indicates that the network with **RGB+TimeDifference** as input tends to learn interactions between the human body and the environment to understand falls.

Task 3: Investigating the human body-centered pattern. Based on the results in Task 1 and Task 2, we conclude that the convolutional network tends to learn body-centered patterns for fall understanding. Here we perform further investigations based on the *low-level* data splitting, as well as the **RGB+TimeDifference** and **Optical Flow** modalities, which represent body-centered context and body motion, respectively. Afterwards, we fuse the two modalities motivated by the two-stream fusion model proposed in [103]. Specifically, we average the softmax outputs from two streams of CED nets with the same (k, l) values. Figure 4.6 shows the results.

One can see that **Optical Flow** and **RGB+TimeDifference** produce comparable performances according to the box plots, yet the network with the **Optical Flow** modality

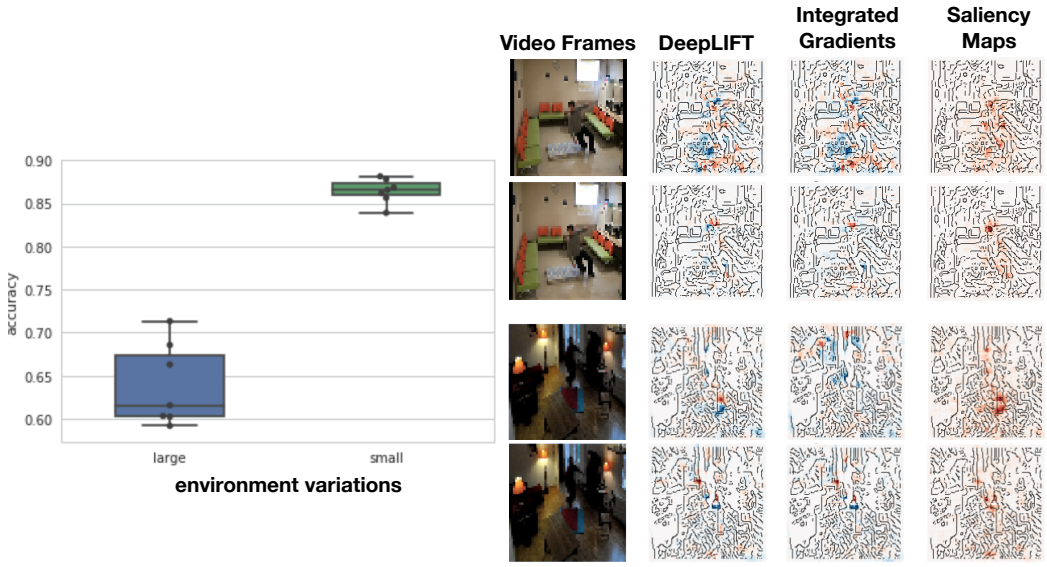

Figure 4.5: Experiment results of Task 2. From left to right: (1) Quantitative results under large environment variations (the *high-level* splits) and small environment variations (the *low-level* splits), with the modality **RGB+TimeDifference**. (2) attribution maps from two testing samples are shown. The first two rows compare the large and small evaluation settings on the same frame in *coffee room*, respectively. The last two rows show another comparison on the same frame in *home*.

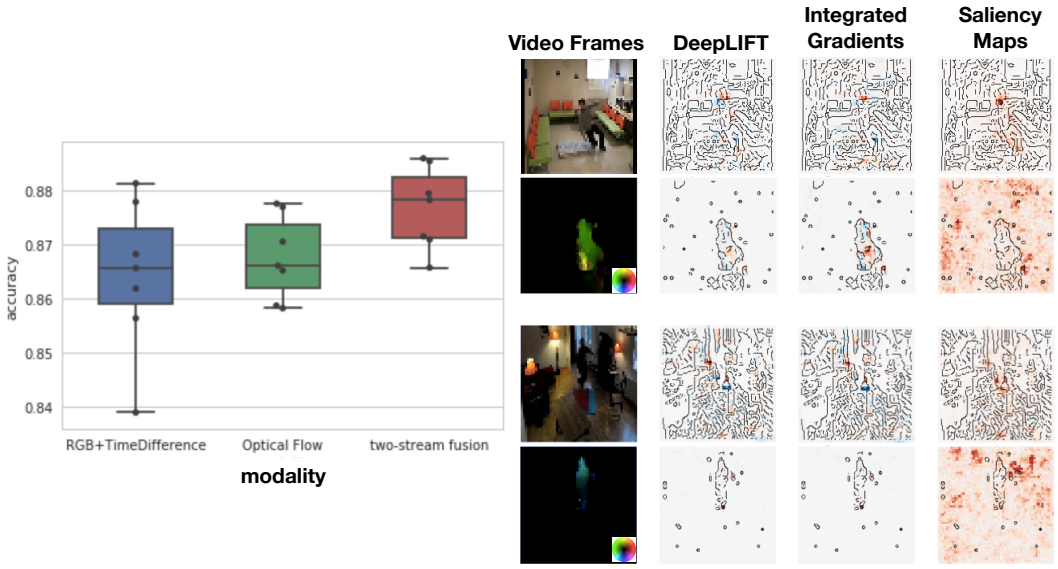

Figure 4.6: Experiment results of Task 3. From left to right: (1) Quantitative results of different modalities under small environment variations (the *low-level* splits). (2) Examples of attribution maps of the two modalities are presented. In particular, the optical flow is visualized using the color coding scheme attached at the bottom-right corner.

behaves more stable than the other case. The fusion results outperform individual modalities. Additionally, from the attribution maps of optical flows we can see that the influential pixels are within the contour of the human body, in contrast to the attribution maps of **RGB+TimeDifference**. Noticeably, one can note that the saliency map is not reliable for **Optical Flow**, since the saliency values are computed as the derivatives of the output w.r.t. the input. Then, zero-value input can cause numerical problems.

A probable reason of the stable performance with **Optical Flow** is that human body motion can represent falls more robustly than the body-centered context, which can be easily influenced by the static background information. The superior performance of modality fusion can indicate that body-centered context and body motion are complementary. The complementary property can also be viewed from the attribution maps, since the influential pixels are at different locations.

Task 4: Investigating the influence of body pose information. Since it is reported in the literature that body pose information from motion capture devices can reliably detect falls [169] [170], here we investigate the influence of 2D pose features, which are estimated from video frames. The pose feature is represented by heat maps, and is extracted using the pre-trained model proposed by [180] [172], which is a deep neural network to regress 2D body joint heat maps from RGB images. The evaluation is based on the *low-level* splitting, as well as the modalities of **Pose**, **Optical Flow** and **Pose+Optical Flow**. The results are shown in Figure 4.7.

One can see that the pose information leads to inferior performances, and also deteriorates the performances of **Optical Flow** when combing flow and pose information. On the right of Figure 4.7, one can find that the influential pixels on the pose heat maps mainly locate at the positions of the non-zero pose scores. Moreover, from the third row on the right of Figure 4.7, one can see that the pose estimation is not always reliable. Thus, we conclude that incorrect pose estimation can degrade fall understanding from videos, although the pose estimation method is state-of-the-art in several public benchmarks. We suggest *not* to use pose features estimated by deep neural networks from RGB images for fall detection.

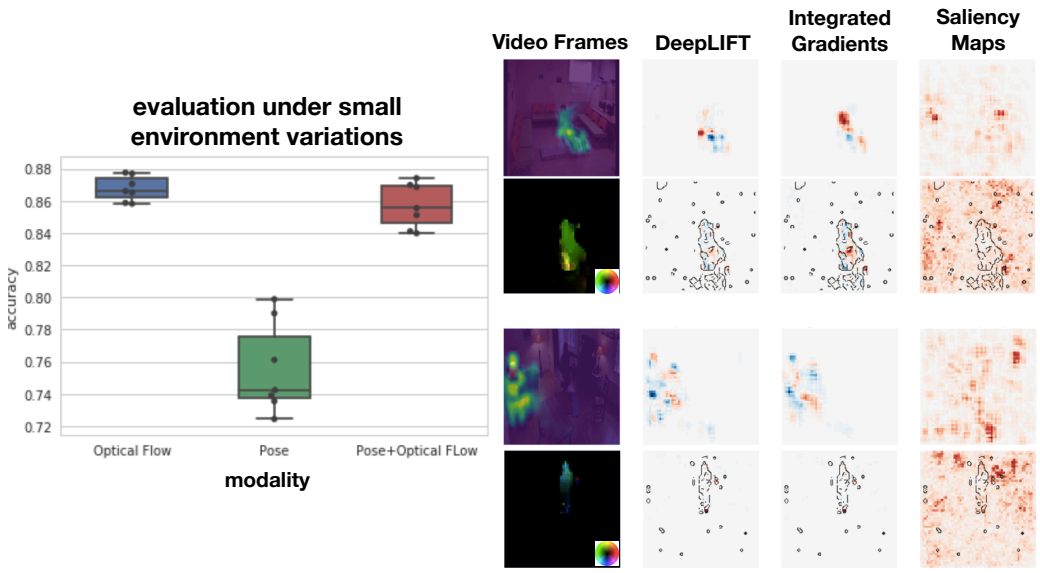

Figure 4.7: Experiment results of Task 4. From left to right: (1) Quantitative results presented by box plots. (2) The attribution maps of pose and optical flow modalities. The selected frames are the same with previous figures. The pose heat map, in which the value increases from blue to yellow, is overlaid with the RGB image only for visualization. The RGB image is not input to the net.

4.2.3 Summary

Based on different types of input modalities and dataset splits, our empirical studies show several influential factors of the model performances. In particular, we find that: (1) The network tends to learn body-centered patterns, but cannot eliminate the influence of static background information, leading to poor cross-environment generalization. (2) Training samples captured from the testing environment can considerably improve the performance, and encourage the network to encode body-centered context, for which the most influential pixels are located around the body contour. (3) The human body motion contains representative features of falls robust to environment changes, and influences on fall recognition in a complementary manner with the body-centered context. (4) Incorrect pose information can degrade the performances heavily. At the current stage, body pose estimation is a challenging task by itself, and the performances are not guaranteed.

Therefore, for detecting falls from videos in practice, we suggest: (1) Including videos captured in the test environments to train the network. (2) Getting rid of information from the static background, and using body-centered features to detect falls. (3) Using multi-modal inputs, in particular, using the fuse of RGB frames, frame time differences, and optical flow. (4) Being careful to use pose information derived by deep neural networks to detect falls. For safety, it is better not to use them.

4.3 Local Temporal Bilinear Pooling

In recent years, deep convolutional nets are widely employed for supervised temporal action parsing. For example, the method proposed in [57] and [59] first extracts frame-wise feature vectors via a spatial convolutional network, and then assigns action labels to individual frames via a temporal convolutional encoder-decoder (TCED) architecture. In these deep neural networks, local temporal pooling, e.g. max pooling or average pooling, is an essential step to aggregate low-level features within temporal neighborhoods to form action patterns, which are able to effectively represent an action spanning several frames.

While being straightforward, a notable caveat is that the first-order pooling operation, e.g. max pooling and average pooling, embedded between convolutional layers ignores high-order temporal structures. A first-order pooling method, e.g. average pooling, computes the mean of the features in a temporal neighborhood to represent that neighborhood, and hence cannot be used to differentiate two actions with identical first-order statistics but different second-order ones. One the other hand, a bilinear pooling method computes the second-order information, e.g. the covariance of the features, to represent that temporal neighborhood, so that the drawbacks of the first-order pooling are overcome. Taking grasping object by hand as an example, when the feature vector of each frame is the concatenation of 3D positions of the finger tips, the first-order pooling, e.g. average pooling, on several consecutive frames yields the hand position, and hence tells where to grasp the object. In parallel, the second-order information can indicate the finger scatter, and hence tells how to grasp the object. Thus, different orders of information are rather independent and complementary to precisely describe an action. Without the second-order information, it is hardly able to distinguish whether to grasp a coin or a book at the same position. Motivated by such fact, as well as several recent studies showing that bilinear pooling consistently outperforms first-order pooling on fine-grained tasks (e.g. [62, 69, 75]), we aim at introducing bilinear pooling into the TCED network as a layer, so as to extract second-order (even higher-order) information to produce better fine-grained action parsing results.

However, introducing bilinear pooling into the TCED network is highly non-trivial, which requires to overcome drawbacks of the conventional bilinear pooling (see Sec. 4.3.2): (1) The conventional bilinear pooling is designed for visual classification, e.g. [65]. Thus, the bilinear pooling aggregates all the features globally to yield a single action pattern, which is not suitable for temporal semantic segmentation. (2) The conventional bilinear pooling lifts the dimension of feature from d to d^2, causing parameter proliferation in the neural network, curse-of-dimensionality and expensive computational cost. To overcome the first drawback, we perform local temporal pooling to aggregate features in local temporal neighbors, like the max pooling operation in TCED. To overcome the second drawback, we design bilinear forms according to reproducing kernel Hilbert space (RKHS) theories

[129], so as to reduce the dimensionality of second-order features while preserving the representativeness.

Following the motivations above, we investigate local temporal bilinear pooling from two perspectives: (1) The first perspective is on *statistics*, especially on how to compute the covariance matrix or the Sums-of-Squares-and-Cross-Products (SSCP) matrix. A first-order pooling method, e.g. average pooling or max pooling, computes the mean/element-wise max of the features in a temporal neighborhood to represent that neighborhood, while a bilinear pooling method computes the second-order information, e.g. the covariance of the features, to represent that temporal neighborhood. Such statistics-based second-order information is straightforward, but its dimensionality is quadratic to the first-order feature, causing curse-of-dimensionality and expensive computation cost. Based on RKHS theories, we reduce the feature dimension by half without information loss. (2) The second perspective is on *multilinear algebra*, especially on three-way tensor product (see Eq. (4.24)). Given a three-way tensor and two input vectors, the output vector of the tensor product can incorporates second-order information if the three-way tensor has certain structures. Compared to the second-order statistics, such tensor product operation produces feature vectors of approximated second-order information and lower dimensionality, which is therefore computationally easier and more suitable to be used in a large network architecture (e.g. [60]).

In the following, we first introduce preliminaries of temporal action parsing networks and conventional bilinear pooling in the literature, and then present our own methods.

4.3.1 Preliminaries: Temporal Action Parsing Networks

Temporal convolutional encoder-decoder (TCED). Here we introduce the TCED network proposed by [59], which is a special network architecture presented in Sec. 4.1.2. The TCED network takes a temporal sequence of feature vectors, and assigns an action label to each input feature vector. It comprises a temporal encoder, a temporal decoder, and a fully connected module to generate frame-wise action labels. The encoder consists of 2 consecutive "non-causal 1D temporal convolution $\rightarrow$ max pooling" modules. After each encoder module, the temporal resolution is halved. The decoder

has a symmetric structure with the encoder, which consists of 2 consecutive "upsampling $\rightarrow$ non-causal 1D temporal convolution" modules, in which the upsampling performs nearest-neighbor interpolation. After each decoder module, the temporal resolution is doubled. The fully connected module incorporates a time-distributed fully connected layer to perform linear transformation at each time instant. Then each output is passed to a softmax function to fit the ground truth one-hot encoded action label.

Multi-stage temporal encoder-decoder network (MS-TCN). The MS-TCN network is an extension of TCED, and is proposed by [60]. Taking a time sequence of frame-wise feature vectors as input, such network performs non-causal dilated temporal convolution in multiple stages, without pooling/upsampling layers to change the temporal resolution. In our study, we use the default network architecture in [60]. It has 4 stages, and each stage has the structure of "linear projection $\rightarrow$ 10 dilated residual layers $\rightarrow$ linear projection". The first linear projection is to project the input features to a latent space, and the second linear projection is to project features in the latent space to the output space, the dimension of which equals to the number of action classes. The dilated residual layer has a dilated 1D temporal convolution layer, a linear projection layer, and a residual connection, which can be formulated as

$$z = ReLU(W_1 * x + b_1) \tag{4.4}$$

$$y = x + W_2 z + b_2, \tag{4.5}$$

in which W_1 denotes the weights in the dilated temporal convolution, W_2 denotes the weights in the linear projection, b_1 and b_2 denote the bias, x and y denote the input and the output of the dilated residual layer. In addition, the 10 dilated residual layers have fixed convolution kernel size equal to 3, but have increasing dilation factors $1, 2, 4, ..., 512$. One notes that the dilated 1D convolution with (dilation=d, kernel size=3) has the receptive field $2d + 1$, and the convolution computation only involves three feature vectors, i.e. the two feature vectors in the two ends of the receptive field, and the feature vector in the middle. Therefore, the dilated 1D temporal convolution has a much larger receptive field,

but an equivalent computational cost with the standard 1D temporal convolution. Dilated 1D convolution with the dilation factor 1 is equivalent to the standard 1D convolution, and dilated 1D convolution with the dilated factor 512 has the receptive field 1025, meaning this temporal convolution covers 1025 frames.

One can see [60, Figure 1] for illustration of MS-TCN. According to their experimental results, MS-TCN outperforms other state-of-the-art deep neural networks on several benchmarks.

4.3.2 Preliminaries: Bilinear Pooling

Since neither TCED nor MS-TCN is able to capture second-order information, here we describe some preliminary knowledge about bilinear pooling, which can work as an intermediate layer in a deep neural network, and extracts second-order information for fine-grained action parsing.

Bilinear pooling from the statistics perspective. Given a set of generic feature vectors with $\boldsymbol{x} \in \mathcal{X}$, such as a set of local spatial image descriptors and a set of frame features, the conventional bilinear pooling [62, 64, 65, 66] can be given by

$$\mathcal{B}(\mathcal{X}) = vec\left(\frac{1}{|\mathcal{X}|}\sum_{\boldsymbol{x}\in\mathcal{X}} \boldsymbol{x}\otimes\boldsymbol{x}\right), \tag{4.6}$$

where $\otimes$ denotes the vector outer product, $|\cdot|$ denotes the cardinality of the feature set, and $vec(\cdot)$ denotes tensor vectorization. In this case, the bilinear composition computes the SSCP matrix of the feature set with the consideration of feature channel correlations. One can note that when $\boldsymbol{x} \in \mathbb{R}^d$, the bilinear pooling lifts the dimension and leads to $\mathcal{B}(\mathcal{X}) \in \mathbb{R}^{d^2}$. In many cases for simplicity, only the diagonal entries of $\mathcal{B}(\mathcal{X})$ are used for second-order information extraction, which is equivalent to $\boldsymbol{x} \circ \boldsymbol{x}$ with $\circ$ being the Hadamard product.

Bilinear pooling from the multilinear algebra perspective. Specifically, given two generic feature vectors $\boldsymbol{x} \in \mathbb{R}^{D_X}$ and $\boldsymbol{y} \in \mathbb{R}^{D_Y}$, the bilinear model based on high-order tensor multiplication [183] is given by:

$$\boldsymbol{z} = \boldsymbol{\mathcal{T}} \times_1 \boldsymbol{x} \times_2 \boldsymbol{y}, \tag{4.7}$$

where $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{D_X \times D_Y \times D_Z}$ is a three-way tensor, and the operations $\times_1$ and $\times_2$ are the mode-1 and mode-2 multiplication, respectively[2]. The tensor $\boldsymbol{\mathcal{T}}$ determines how information in the output $\boldsymbol{z} \in \mathbb{R}^{D_Z}$ is constructed from interactions between the elements in $\boldsymbol{x}$ and $\boldsymbol{y}$. Though being a powerful scheme for information fusion, such model tends to suffer from *curse of dimensionality* and *intractable computation* [184]. Specifically, $\dim(\boldsymbol{\mathcal{T}})$ grows exponentially with the number of available channels. For the bilinear models in Eq. (4.7), the number of free parameters grows cubically $\mathcal{O}(D_X D_Y D_Z)$ with respect to feature dimensions, which is prohibitive even for moderate number of features, hence limiting the representation power of the model.

Their connections. Such two perspectives are equivalent if specific structure of $\boldsymbol{\mathcal{T}}$ in Eq. (4.7) is applied. For example, when $D_X = D_Y = D_Z = D$ and $\boldsymbol{\mathcal{T}}$ is a three-way identity tensor, we simply have $\boldsymbol{z} = \boldsymbol{x} \circ \boldsymbol{y}$ with $\circ$ being the Hadamard product (element-wise product). When the mode-3 matricization of $\boldsymbol{\mathcal{T}}$ is an identity matrix of size $D_X D_Y$, we have $\boldsymbol{z} = vec(\boldsymbol{x} \otimes \boldsymbol{y})$, i.e., the vectorization of the outer product between the two input features.

4.3.3 Preliminaries: Normalization

The bilinear forms, such as Eq. (4.6) and Eq. (4.7), can be applied in a generic deep neural network as an intermediate layer, e.g. between two 1D convolution layers to extract second-order information. Due to the tensor product (or vector outer product), small values become smaller and large values become larger as the data flows from the

[2]Given a three-way tensor $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{D_X \times D_Y \times D_Z}$, the mode-1 multiplication with a vector $\boldsymbol{x}$, i.e. $\boldsymbol{\mathcal{T}} \times_1 \boldsymbol{x}$, yields a 2D matrix with the elements $\{\sum_i t_{ijk} x_i\}_{jk}$. The mode-2 multiplication works in an equivalent manner. Therefore, in Eq. (4.7), it has $z_k = \sum_{i,j} t_{ijk} x_i y_j$.

network bottom to up, which can lead to diverging spectra in the bilinear forms and very sparse features before the final classification layer. Here we present three normalization methods that can address such problems.

l_2 normalization. We can apply l_2 normalization on the output of bilinear pooling. Since the l_2 norm of a vectorized matrix is equivalent to its Frobenius norm and also equivalent to the Frobenius norm of the singular value matrix after SVD, the l_2 normalization on the vectorized bilinear form can constrain the matrix spectra between 0 and 1, and hence effectively eliminates the diverging spectra problem.

Normalized ReLU. [59] proposes a normalized ReLU activation function, which allows fast convergence and yields superior results to other activation functions. Given an input vector $\boldsymbol{x}$, normalized ReLU can be given by

$$\sigma(\boldsymbol{x}) = \textbf{NReLU}(\boldsymbol{x}) = \frac{ReLU(\boldsymbol{x})}{max(ReLU(\boldsymbol{x})) + \epsilon}, \tag{4.8}$$

where **NReLU** stands for normalized ReLU, ϵ is a small positive constant and the $max(\cdot)$ operation selects the maximal entry in the vector $\boldsymbol{x}$. Since the Frobenius norm is bounded from above by the max norm multiplying with a scalar [185], **NReLU** is also able to constrain the matrix singular values and hence eliminates the diverging spectra issue. Nevertheless, it can lead to sparse features.

Regularized power normalization. Power normalization is an effective trick to overcome sparse features and improve the performance [62, 67], which can be formulated as $\text{sign}(x) \cdot \sqrt{|x|}$. However, when using power normalization in intermediate layers of a neural net, it can cause gradient explosion during back-propagation, since the gradient goes to infinity when encountering 0 values. Therefore, we propose a regularized version, and use it as an activation function after each temporal convolution layer, so that features in the network are always densified. The formula is given by

$$\sigma(x) = \textbf{RPN}(x) = \text{sign}(x) \cdot \left(\sqrt{|x| + \theta^2} - \sqrt{\theta^2} \right), \tag{4.9}$$

in which **RPN** stands for *regularized power normalization* and θ is can be a learnable parameter. As $\theta \to 0$, the **RPN** function converges to the standard power normalization.

4.3.4 Local Temporal Bilinear Pooling: A Statistical Perspective

Our statistics-based bilinear pooling method [3] is modified from the global pooling Eq. (4.6). In contrast to the global pooling, which destroys local structures, we use a local temporal neighbourhood in order to preserve the local temporal structure. Specifically, given a temporal sequence of feature vectors $\boldsymbol{X} = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_T\}$ with $\boldsymbol{x}_t \in \mathbb{R}^d$ for $t \in 1, 2, ..., T$, the local temporal bilinear composition is given by

$$\mathcal{B}(\boldsymbol{x}_t) = vec\left(\frac{1}{|\mathcal{N}(t)|} \cdot \sum_{\tau \in \mathcal{N}(t)} \boldsymbol{x}_\tau \otimes \boldsymbol{x}_\tau \right), \tag{4.10}$$

where $\mathcal{N}(t)$ denotes the *local* temporal neighborhood set centered at the time instant t. One can note that such operation is non-causal. In addition, we improve the bilinear pooling in Eq. (4.10) by the following two aspects.

- **Decoupling First and Second-order Information.** Inspired by a physical fact that the position and the velocity of an object in motion indicate different dynamical variables, we consider to separate first and second-order components from the bilinear form (Eq. (4.10)) to describe the action via separate attributions. In this case, we propose a decoupled bilinear form, which is denoted as $\mathcal{B}_d(\cdot)$. Provided a time sequence of d-dimensional feature vectors $\{\boldsymbol{x}_1, ..., \boldsymbol{x}_T\}$, the first-order component $\boldsymbol{\mu}$, the second-order component $\boldsymbol{\Sigma}$ and $\mathcal{B}_d(\cdot)$ are given by

[3] Our statistics-based bilinear pooling study, as well as the corresponding experiments in Sec. 4.4.3, are published in [6] © 2019 IEEE (textual content © 2011 IEEE).

$$\boldsymbol{\mu}_t = \frac{1}{|\mathcal{N}(t)|} \cdot \sum_{\tau \in \mathcal{N}(t)} \boldsymbol{x}_\tau, \tag{4.11}$$

$$\boldsymbol{\Sigma}_t = \frac{1}{|\mathcal{N}(t)|} \cdot \sum_{\tau \in \mathcal{N}(t)} (\boldsymbol{x}_\tau - \boldsymbol{\mu}_t) \otimes (\boldsymbol{x}_\tau - \boldsymbol{\mu}_t) \text{ and} \tag{4.12}$$

$$\mathcal{B}_d(\boldsymbol{x}_t) = \left(\boldsymbol{\mu}_t^T, vec(\boldsymbol{\Sigma}_t)\right)^T, \tag{4.13}$$

in which one can note $\mathcal{B}_d(\boldsymbol{x}_t) \in \mathbb{R}^{d(d+1)}$. Since the first-order component is equivalent to the mean and the second-order component is equivalent to the covariance, such decomposed bilinear form can precisely describe a Gaussian distribution. Additionally, we denote the bilinear form in Eq. (4.10) as $\mathcal{B}_c(\cdot)$, which means “coupled”.

- **Adapting local statistics to data.** When the local statistics is more complex than Gaussian distribution, only using mean and covariance to describe the statistics is not sufficient. Rather than applying higher-order statistics like in [90, 91], we stay in second-order statistics to retain computational cost low. Since the averaging operation in Eq. (4.10) and Eq. (4.11) can be regarded as convolution by a box filter, we generalize it to convolution by a learnable filter. Thus, the local statistics is adaptive to the data and the network objective. Specifically, for the coupled bilinear form the learnable version is given by

$$\mathcal{B}_c(\boldsymbol{x}_t) = vec\left(\sum_{\tau \in \mathcal{N}(t)} \omega_\tau \boldsymbol{x}_\tau \otimes \boldsymbol{x}_\tau\right), \tag{4.14}$$

where the filter weights $\{\omega_\tau\}$ are shared by all temporal neighbourhood sets, i.e. $\mathcal{N}(t)$ with $t = 1, 2, ..., T$. The filter weight ω_τ represents the contribution of the input feature vector at the frame τ to the bilinear pooling output, and works in a similar manner to temporal convolution. Noticeably, the weight value can be negative.

Likewise, for the decoupled bilinear form, the learnable version is given by

$$\boldsymbol{\mu}_t = \sum_{\tau \in \mathcal{N}(t)} p_\tau \boldsymbol{x}_\tau, \tag{4.15}$$

$$\boldsymbol{\Sigma}_t = \sum_{\tau \in \mathcal{N}(t)} q_\tau (\boldsymbol{x}_\tau - \boldsymbol{\mu}_t) \otimes (\boldsymbol{x}_\tau - \boldsymbol{\mu}_t) \text{ and} \tag{4.16}$$

$$\mathcal{B}_d(\boldsymbol{x}_t) = \Big(\boldsymbol{\mu}_t^T, vec(\boldsymbol{\Sigma}_t)\Big)^T, \tag{4.17}$$

where the learnable filter weights $\{p_\tau\}$ and $\{q_\tau\}$ are shared by all temporal neighborhood sets.

Intuitively, the major difference between $\mathcal{B}_c(\cdot)$ and $\mathcal{B}_d(\cdot)$ is whether the extracted second-order information is centered or not. Depending on the input data to the bilinear pooling, such difference can be essential or not. If the input samples to $\mathcal{B}_c(\cdot)$ and $\mathcal{B}_d(\cdot)$ are already centered, $\mathcal{B}_c(\cdot)$ and $\mathcal{B}_d(\cdot)$ are expected to have comparable performance. Even, $\mathcal{B}_c(\cdot)$ is better, since the $\boldsymbol{\mu}_t$ in $\mathcal{B}_d(\cdot)$ is close to zero and hence somehow redundant. If the input samples to $\mathcal{B}_c(\cdot)$ and $\mathcal{B}_d(\cdot)$ are not well centered, for example, the input samples to the bilinear pooling are the outputs from a convolutional layer with ReLU and hence all input samples are positive, then the difference is essential. The most principal eigenvalue/eigenvector of $\mathcal{B}_c(\cdot)$ is dominated by the average locations of input samples, leading to degraded second-order information representation, i.e. how the input samples are scattered. On the other hand, the component $\boldsymbol{\Sigma}_t$ in $\mathcal{B}_d(\cdot)$ can fully reflect how the input samples are scattered. The input sample locations are reflected by $\boldsymbol{\mu}_t$ additionally. Therefore, in our study, we use both $\mathcal{B}_c(\cdot)$ and $\mathcal{B}_d(\cdot)$ in our experiments for fine-grained action parsing (see Sec. 4.4).

Low-dimensional Representation. Given an arbitrary feature vector sequence, the bilinear forms $\mathcal{B}_c$ and $\mathcal{B}_d$ can capture local temporal statistics which are adaptive to the data. However, the feature dimension is considerably increased. Specifically, given $\boldsymbol{x}_t \in \mathbb{R}^d$, we have $\mathcal{B}_c(\boldsymbol{x}_t) \in \mathbb{R}^{d^2}$ and $\mathcal{B}_d(\boldsymbol{x}_t) \in \mathbb{R}^{d(d+1)}$. To address such issue, we propose lower-dimensional alternative representations to the explicit bilinear forms defined in Eq. (4.14) and Eq. (4.17). Consequently, our method is *exact*, which means it introduces

neither information loss as in approximation methods nor computational cost as in PCA or SVD.

To find such lower-dimensional representations, we first show that $\mathcal{B}_c(\cdot)$ and $\mathcal{B}_d(\cdot)$ are feature mappings associated with reproducing kernel Hilbert spaces (RKHSs) [186], for which the kernels are second-order homogeneous and inhomogeneous polynomials, respectively. Such property can be extended to arbitrary p-th order polynomials. One can refer to more details in [187, Chapter 3].

Proposition 1. *Given* $\{\boldsymbol{x}_1, ..., \boldsymbol{x}_T\}$*, we have*

$$\langle \mathcal{B}_c(\boldsymbol{x}_i), \mathcal{B}_c(\boldsymbol{x}_j) \rangle_{\mathbb{R}^{d^2}} = \sum_{\tau \in \mathcal{N}(i)} \sum_{\tau' \in \mathcal{N}(j)} \omega_\tau \omega_{\tau'} \langle \boldsymbol{x}_\tau, \boldsymbol{x}_{\tau'} \rangle^2_{\mathbb{R}^d} , \tag{4.18}$$

and

$$\begin{aligned} &\langle \mathcal{B}_d(\boldsymbol{x}_i), \mathcal{B}_d(\boldsymbol{x}_j) \rangle_{\mathbb{R}^{d(d+1)}} = \langle \boldsymbol{\mu}_i, \boldsymbol{\mu}_j \rangle_{\mathbb{R}^d} \\ &+ \sum_{\tau \in \mathcal{N}(i)} \sum_{\tau' \in \mathcal{N}(j)} q_\tau q_{\tau'} \langle \boldsymbol{x}_\tau - \boldsymbol{\mu}_i, \boldsymbol{x}_{\tau'} - \boldsymbol{\mu}_j \rangle^2_{\mathbb{R}^{d^2}} , \end{aligned} \tag{4.19}$$

in which the notations are referred to the definitions in Eq. (4.14) *and Eq.* (4.17)*.*

Proof. For the coupled bilinear composition, we have

$$\begin{aligned} &\langle \mathcal{B}_c(\boldsymbol{x}_i), \mathcal{B}_c(\boldsymbol{x}_j) \rangle_{\mathbb{R}^{d^2}} \\ &= \left\langle \sum_{\tau \in \mathcal{N}(i)} vec\left(\omega_\tau \boldsymbol{x}_\tau \otimes \boldsymbol{x}_\tau\right), \sum_{\tau' \in \mathcal{N}(j)} vec\left(\omega_{\tau'} \boldsymbol{x}_{\tau'} \otimes \boldsymbol{x}_{\tau'}\right) \right\rangle \\ &= \sum_{\tau \in \mathcal{N}(i)} \sum_{\tau' \in \mathcal{N}(j)} \omega_\tau \omega_{\tau'} \left\langle vec\left(\boldsymbol{x}_\tau \otimes \boldsymbol{x}_\tau\right), vec\left(\boldsymbol{x}_{\tau'} \otimes \boldsymbol{x}_{\tau'}\right) \right\rangle \\ &= \sum_{\tau \in \mathcal{N}(i)} \sum_{\tau' \in \mathcal{N}(j)} \omega_\tau \omega_{\tau'} \langle \boldsymbol{x}_\tau, \boldsymbol{x}_{\tau'} \rangle^2_{\mathbb{R}^d} . \end{aligned} \tag{4.20}$$

For the decoupled bilinear composition, we have

$$\langle \mathcal{B}_d(\boldsymbol{x}_i), \mathcal{B}_d(\boldsymbol{x}_j) \rangle_{\mathbb{R}^{d(d+1)}} = \langle \boldsymbol{\mu}_i, \boldsymbol{\mu}_j \rangle_{\mathbb{R}^d} + \langle vec(\boldsymbol{\Sigma}_i), vec(\boldsymbol{\Sigma}_j) \rangle_{\mathbb{R}^{d^2}}. \tag{4.21}$$

Thus, one can easily derive the formula in the proposition. □

One can see from Proposition 1 that the inner product of $\mathcal{B}_c(\cdot)$ can be expressed in terms of the 2nd-degree homogeneous polynomial kernel $k(\mathbf{x}, \mathbf{x}') = \langle \mathbf{x}, \mathbf{x}' \rangle^2$. In general, the dimension of $\mathcal{B}_c(\cdot)$ increases *exponentially* with the degree of the polynomial kernel, making it less practical when used explicitly in a deep neural net. Motivated by the fact that for a specific kernel $k(\cdot, \cdot)$, the associated feature mapping $\phi : \boldsymbol{X} \rightarrow \mathcal{H}$ is not unique, we derive a feature mapping that corresponds to the same kernel as $\mathcal{B}_c(\cdot)$, but has lower dimension. For example, provided the vector $\boldsymbol{x} = (x_1, x_2)^T$, and the polynomial kernel $k(\boldsymbol{x}_1, \boldsymbol{x}_2) = \langle \boldsymbol{x}_1, \boldsymbol{x}_2 \rangle^2 = \langle \phi(\boldsymbol{x}_1), \phi(\boldsymbol{x}_2) \rangle$, we can derive $\phi(\boldsymbol{x}) = (x_1^2, x_1x_2, x_2x_1, x_2^2)^T$, or $\phi(\boldsymbol{x}) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)^T$, in which the latter one has lower dimensionality. Therefore, the proposed method reduces the number of parameters to be learned without sacrificing the representativeness. In particular, we show that:

Proposition 2. *Let $\mathcal{B}_c(\boldsymbol{x}) \in \mathbb{R}^{d^2}$ be the bilinear composition and $\phi_c(\boldsymbol{x}) \in \mathbb{R}^{\frac{d(d+1)}{2}}$ a feature mapping defined by*

$$\phi_c(\boldsymbol{x}) = (\underbrace{x_1^2, ..., x_d^2}_{d \text{ terms}}, \underbrace{\sqrt{2}x_1x_2, \sqrt{2}x_1x_3, ..., \sqrt{2}x_{d-1}x_d}_{C(d,2) \text{ terms}})^T. \tag{4.22}$$

Then, it follows that for any $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^d$, it has

$$\langle \mathcal{B}_c(\boldsymbol{x}), \mathcal{B}_c(\boldsymbol{x}') \rangle_{\mathbb{R}^{d^2}} = \langle \phi_c(\boldsymbol{x}), \phi_c(\boldsymbol{x}') \rangle_{\mathbb{R}^{\frac{d(d+1)}{2}}}. \tag{4.23}$$

Equivalently, the second-order component defined in Eq. (4.17) *has a lower-dimensional alternative, so that $\mathcal{B}_d(\boldsymbol{x}) \in \mathbb{R}^{d(d+1)}$ can be replaced by $\phi_d(\boldsymbol{x}) \in \mathbb{R}^{\frac{d(d+3)}{2}}$.*

Due to the commutative property of tensor product, the above proposition can be proved by reformulating the polynomials in Eq. (4.20).

Proposition 2 shows that $\mathcal{B}_c(\cdot)$ and $\phi_c(\cdot)$ are equivalent, in the sense that the corresponding kernel is the same (see Eq. (4.23)). The advantage of using $\phi_c(\cdot)$ instead of $\mathcal{B}_c(\cdot)$ is that it has much lower dimension. For example, if each feature vector in the input sequence is 128-dimensional, $\mathcal{B}_c$ is 16384-dimensional and $\mathcal{B}_d$ is 16512-dimensional. On the other hand, the alternative feature representations ϕ_c is 8256-dimensional and ϕ_d is 8384-dimensional, approximately halving the dimensionality without losing information and without introducing extra computation.

4.3.5 Local Temporal Bilinear Pooling: A Multilinear Algebraic Perspective

Although our statistics-based bilinear pooling method can extract 2nd-order information with half dimensionality of the full bilinear form, the computational cost is still high, i.e., $\mathcal{O}(d^2)$ w.r.t. the input feature dimension d, and hence is difficult to be used in a larger architecture than the TCED network [59]. In addition, with our statistics-based bilinear model, one cannot extract higher-order (third-order and higher) information, which is more discriminative and yield better performance [91, 90].

Although modeling high-order feature interaction has many successful applications, most existing methods have a critical drawback: *The computational cost increases exponentially with respect to the information order and the input feature dimension*. Let us take bilinear pooling for second-order information extraction as an example. Given two feature vectors $\boldsymbol{x} \in \mathbb{R}^{D_X}$ and $\boldsymbol{y} \in \mathbb{R}^{D_Y}$, a generic bilinear model [183, Eq. (2)] is given by:

$$\boldsymbol{z} = \boldsymbol{\mathcal{T}} \times_1 \boldsymbol{x} \times_2 \boldsymbol{y}, \tag{4.24}$$

where $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{D_X \times D_Y \times D_Z}$ is a three-way tensor, and the operations $\times_1$ and $\times_2$ are the mode-1 and mode-2 multiplication, respectively. Despite being a powerful scheme, such a model tends to suffer from the *curse of dimensionality* and *intractable computation* [184]. Specifically, the number of free parameters in $\boldsymbol{\mathcal{T}}$ grows cubically as the feature dimensionality, i.e. $\mathcal{O}(D_X D_Y D_Z)$, hence limiting its use in large-scale scenarios.

To reduce the complexity, a common solution is to impose specific assumptions on the structure of $\mathcal{T}$ in Eq. (4.24). In the work of [80, 183, 71], low-rank assumption on $\mathcal{T}$ is employed. In the work of [69], count sketch is used to extract approximated second-order information, which can be represented by a shorter feature vector than the full second-order information. Although such conventional bilinear pooling methods can extract second-order information effectively, extracting higher-order information requires extra operations [91] and may further increase the computational cost.

Here, we aim at deriving a simple bilinear model, with which changing the order of information does not involve extra computation. Therefore, our investigations have two aspects: (1) From the multilinear perspective, we assume that each frontal slice of $\mathcal{T}$ — but not $\mathcal{T}$ itself — is a low-rank matrix. Then we derive a new bilinear form that first reduces the feature dimension and then performs vector outer product. Different from [80] and [183], that first lift the feature dimension and then perform Hadamard product, our method has runtime complexity $\mathcal{O}(D\sqrt{d}+d)$ and space complexity $\mathcal{O}(D\sqrt{d})$ compared to $\mathcal{O}(Dd+d)$ and $\mathcal{O}(Dd)$ in their works[4]. Thus, our bilinear operation is lightweight and can be employed in deep neural networks as intermediate layers in an efficient manner. (2) Rather than learning the model parameters from data, the entries of each frontal slice are determined by random projection matrices drawn from a specific distribution. Our key insight is that, while the low-rank structure allows a reduction in the number of parameters, the loss in representational power is compensated by the random projection matrices. In particular, we show that when these random matrices are sampled with different underlying distributions, the model approximates feature maps of different reproducing kernel Hilbert spaces (RKHSs). For example, when they are sampled from the Rademacher distribution, the model approximates the multiplication of linear kernels (cf. Theorem 1). When they are sampled from the Gaussian distribution with orthogonality constraints, the model approximates multiplication of Gaussian kernels (cf. Theorem 2). Hence, we can explicitly manipulate the model capacity without sacrificing the computational efficiency.

[4] D and d are input and output feature dimensions of the bilinear model, respectively; usually, $D \ll d$.

4.3.5.1 Tensor Frontal Low-rank Approximation

Here we follow the tensor notations in [188]. Eq. (4.24) can be re-written in terms of matrix-vector multiplication as

$$\boldsymbol{z} = \boldsymbol{T}_{(3)}\, vec(\boldsymbol{x} \otimes \boldsymbol{y}), \tag{4.25}$$

where $\boldsymbol{T}_{(3)}$ is the mode-3 matricization of $\boldsymbol{\mathcal{T}}$, $vec(\cdot)$ denotes column-wise vectorization of a matrix, and $\otimes$ denotes vector outer product. In other words, the bilinear operation in Eq. (4.24) is equivalent to first computing the correlation matrix between the two features and then performing linear projection.

It follows from Eq. (4.25) that each entry of the output feature vector $\boldsymbol{z}$ is a weighted sum of all the entries in the correlation matrix $\boldsymbol{x} \otimes \boldsymbol{y}$, i.e.,

$$z_k = \langle vec(\mathcal{T}[:,:,k]), vec(\boldsymbol{x} \otimes \boldsymbol{y}) \rangle = \sum_{i=1}^{D_X} \sum_{j=1}^{D_Y} \mathcal{T}[i,j,k] v_{f(i,j)}, \tag{4.26}$$

where $\boldsymbol{v} := vec(\boldsymbol{x} \otimes \boldsymbol{y})$. If the frontal matrix $\mathcal{T}[:,:,k]$ is a rank-one matrix, i.e., $\mathcal{T}[:,:,k] = \boldsymbol{e} \otimes \boldsymbol{f}$ for some $\boldsymbol{e} \in \mathbb{R}^{D_X}$ and $\boldsymbol{f} \in \mathbb{R}^{D_Y}$, then we can rewrite Eq. (4.26) as $z_k = \langle vec(\boldsymbol{e} \otimes \boldsymbol{f}), vec(\boldsymbol{x} \otimes \boldsymbol{y}) \rangle = \langle \boldsymbol{e}, \boldsymbol{x} \rangle \langle \boldsymbol{f}, \boldsymbol{y} \rangle$. Thus, we define the projection matrices $\boldsymbol{E} = [\boldsymbol{e}_1, \boldsymbol{e}_2, ..., \boldsymbol{e}_M]^T$ and $\boldsymbol{F} = [\boldsymbol{f}_1, \boldsymbol{f}_2, ..., \boldsymbol{f}_N]^T$ for two sets of vectors $\{\boldsymbol{e}_i\}_{i=1}^M \subset \mathcal{X}$ and $\{\boldsymbol{f}_j\}_{j=1}^N \subset \mathcal{Y}$ with $M \leq D_X$ and $N \leq D_Y$. Then, the fusion map $\phi : \mathbb{R}^{D_X} \times \mathbb{R}^{D_Y} \rightarrow \mathbb{R}^{MN}$ can be defined as

$$\boldsymbol{z} := \phi(\boldsymbol{x}, \boldsymbol{y}) = vec\left((\boldsymbol{E}\boldsymbol{x}) \otimes (\boldsymbol{F}\boldsymbol{y})\right). \tag{4.27}$$

If we assume further that $\mathcal{T}[:,:,k]$ is a rank-R matrix, i.e., $\mathcal{T}[:,:,k] = \sum_{r=1}^{R} \boldsymbol{e}_i^r \otimes \boldsymbol{f}_j^r$, we obtain

$$\boldsymbol{z} := \phi(\boldsymbol{x}, \boldsymbol{y}) = vec\left(\sum_{r=1}^{R} (\boldsymbol{E}^r \boldsymbol{x}) \otimes (\boldsymbol{F}^r \boldsymbol{y})\right), \tag{4.28}$$

where $\boldsymbol{E}^r = [\boldsymbol{e}_1^r, \boldsymbol{e}_2^r, ..., \boldsymbol{e}_M^r]^T$ and $\boldsymbol{F}^r = [\boldsymbol{f}_1^r, \boldsymbol{f}_2^r, ..., \boldsymbol{f}_N^r]^T$ for $r = 1, 2, ..., R$. With such low-rank assumption, we avoid computing the high-dimensional correlation matrix $\boldsymbol{x} \otimes \boldsymbol{y}$, which considerably reduces the model parameters from $D_X D_Y D_Z$ to $R(MD_X + ND_Y)$ with a small value of R.

Similar low-rank assumption is also used in [183, 80], in which the two input feature vectors are first projected to a common vector space and then fused via Hadamard product. Assuming the input feature vectors are of the same dimension D and the output feature vector is of dimension d, then such operation requires $\mathcal{O}(Dd + d)$ operations to compute and requires $\mathcal{O}(Dd)$ memory to store. In contrast, our method requires $\mathcal{O}(D\sqrt{d} + d)$ for computation and $\mathcal{O}(D\sqrt{d})$ for storage. Since in practice it normally requires more dimensions to represent a higher-order feature, i.e. $d \gg D$, our method has consistently better runtime (see Table 4.9) and hence is more suitable to be employed in a sophisticated deep neural network as a pooling layer.

4.3.5.2 Random Projection

To compensate for the loss in model capacity caused by the low-rank assumption, we observe that the entries and associated distributions of $\boldsymbol{E}$ and $\boldsymbol{F}$ defined in Eq. (4.27) and Eq. (4.28) can indeed influence the model capacity, without adding or removing learnable parameters, network layers, etc.

Inspired by this observation, we propose to manipulate the model capacity by randomly sampling the entries of $\boldsymbol{E}$ and $\boldsymbol{F}$ from specific distributions and then perform random projection. Unlike an end-to-end learning via back-propagation, our approach provides an alternative way of building expressive representation that is explainable and is simple to use in practice.

Rademacher random projection. Motivated by [189] and [69], we specify model parameters, i.e. the projection matrices $\boldsymbol{E}^r$ and $\boldsymbol{F}^r$ in the bilinear model (4.27) or (4.28), with random samples from the Rademacher distribution. Below we show that the bilinear model given in Eq. (4.27) *unbiasedly* approximates, with high probability, a feature map to a reproducing kernel Hilbert space (RKHS), in which the associated kernel is the multiplication of two linear kernels in $\mathcal{X}$ and $\mathcal{Y}$, respectively.

Theorem 1. *Let $\boldsymbol{E}^r \in \mathbb{R}^{M \times D_X}$ and $\boldsymbol{F}^r \in \mathbb{R}^{N \times D_Y}$ for any $r \in \{1, 2, ..., R\}$ be Rademacher random matrices whose entries are determined by an independent Rademacher random variable $\sigma \in \{-1, 1\}$. For any $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{X}$ and $\boldsymbol{y}_1, \boldsymbol{y}_2 \in \mathcal{Y}$, let $\boldsymbol{z}_1 = \phi(\boldsymbol{x}_1, \boldsymbol{y}_1)$ and*

$\boldsymbol{z}_2 = \phi(\boldsymbol{x}_2, \boldsymbol{y}_2)$ be the output features given by Eq. (4.27)*. Define a kernel function by $k(\boldsymbol{z}_1, \boldsymbol{z}_2) = \langle \boldsymbol{z}_1, \boldsymbol{z}_2 \rangle$, then we have*

$$\mathbb{E}[k(\boldsymbol{z}_1, \boldsymbol{z}_2)] = RMN \langle \boldsymbol{x}_1, \boldsymbol{x}_2 \rangle \langle \boldsymbol{y}_1, \boldsymbol{y}_2 \rangle.$$

Next, we characterize the error of such kernel approximations.

Corollary 1. *Let $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$ be defined as in Theorem 1. Let $k(\boldsymbol{z}_1, \boldsymbol{z}_2) = \frac{1}{R}\langle \boldsymbol{z}_1, \boldsymbol{z}_2 \rangle$. Then, the following inequality holds:*

$$\mathbb{P}\left(|k(\boldsymbol{z}_1, \boldsymbol{z}_2) - \mathbb{E}[k(\boldsymbol{z}_1, \boldsymbol{z}_2)]| > \epsilon\right) \leq 2\exp\left(-\frac{\epsilon^2 MN}{2p^8 \tilde{R}^8}\right), \tag{4.29}$$

for some constants $\epsilon > 0$, and $p \geq 1$, $\tilde{R} \geq 1$, which are independent of the feature dimensions.

Proofs of both results can be found in Appendix 1 and 2. More details on the constants p and $\tilde{R}$ can be found in [189]. To remove the effect of the scaling factors, we rewrite Eq. (4.28) as

$$\boldsymbol{z} = \phi(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{R\sqrt{MN}} \cdot vec\left(\sum_{r=1}^{R} (\boldsymbol{E}^r \boldsymbol{x}) \otimes (\boldsymbol{F}^r \boldsymbol{y})\right). \tag{4.30}$$

Therefore, Eq. (4.30) with *binary tensor entries* is capable of capturing second-order interactions between two features. We refer this bilinear form as “RPBinary” in the experiment section.

Gaussian random projection. To increase the model capacity, a common approach is to apply the nonlinear activation function $\varphi(\cdot)$, which gives

$$\boldsymbol{z} := \phi(\boldsymbol{x}, \boldsymbol{y}) = vec\left(\sum_{r=1}^{R} \varphi(\boldsymbol{E}^r \boldsymbol{x}) \otimes \varphi(\boldsymbol{F}^r \boldsymbol{y})\right). \tag{4.31}$$

Inspired by [190], we consider

$$\boldsymbol{E}^r = \frac{1}{\sigma^r} \boldsymbol{I}_{M \times D_X} \boldsymbol{R}^r \boldsymbol{P}^r, \quad \boldsymbol{F}^r = \frac{1}{\rho^r} \boldsymbol{I}_{N \times D_Y} \boldsymbol{S}^r \boldsymbol{Q}^r \quad \text{with} \quad r = 1, 2, ..., R, \tag{4.32}$$

where $\boldsymbol{R}^r$ and $\boldsymbol{S}^r$ are diagonal matrices with diagonal entries sampled i.i.d. from the chi-squared distributions $\chi^2(D_X)$ and $\chi^2(D_Y)$ with D_X and D_Y degrees-of-freedom, respectively, $\boldsymbol{P}^r$ and $\boldsymbol{Q}^r$ are uniformly distributed random orthogonal matrices[5], and $\boldsymbol{I}_{M\times D_X}$ and $\boldsymbol{I}_{N\times D_Y}$ are identity matrices with the first M and N rows, respectively. Here, $\{\sigma^r\}_{r=1}^R$ and $\{\rho^r\}_{r=1}^R$ are tunable bandwidth parameters.

When the nonlinear function in Eq. (4.31) is given by $\varphi(\boldsymbol{E}\boldsymbol{x}) := \sqrt{1/M}[\sin(\boldsymbol{E}\boldsymbol{x}), \cos(\boldsymbol{E}\boldsymbol{x})]$ and $\varphi(\boldsymbol{F}\boldsymbol{y}) := \sqrt{1/N}[\sin(\boldsymbol{F}\boldsymbol{y}), \cos(\boldsymbol{F}\boldsymbol{y})]$, the resulting representation approximates feature maps to the RKHSs corresponding to a composition of Gaussian kernels (see Appendix 3 for the proof).

Theorem 2. *Let $\boldsymbol{E}^r \in \mathbb{R}^{M\times D_X}$ and $\boldsymbol{F}^r \in \mathbb{R}^{N\times D_Y}$ for any $r \in \{1, 2, ..., R\}$ be random matrices whose entries are determined as in Eq.* (4.32)*. For any $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{X}$ and $\boldsymbol{y}_1, \boldsymbol{y}_2 \in \mathcal{Y}$, let $\boldsymbol{z}_1 = \phi(\boldsymbol{x}_1, \boldsymbol{y}_1)$ and $\boldsymbol{z}_2 = \phi(\boldsymbol{x}_2, \boldsymbol{y}_2)$ be the output features in Eq.* (4.31)*. Define a kernel function $k(\boldsymbol{z}_1, \boldsymbol{z}_2) = \langle \boldsymbol{z}_1, \boldsymbol{z}_2 \rangle$, then we have*

$$\mathbb{E}[k(\boldsymbol{z}_1, \boldsymbol{z}_2)] = \sum_{r=1}^{R} \exp\left(-\frac{\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2^2}{2\sigma_r^2}\right) \exp\left(-\frac{\|\boldsymbol{y}_1 - \boldsymbol{y}_2\|_2^2}{2\rho_r^2}\right) + b, \tag{4.33}$$

where $b := \sum_{r=1}^{R} \sum_{r'=r+1}^{R} \mathbb{E}[\langle \varphi(\boldsymbol{E}^r \boldsymbol{x}_1), \varphi(\boldsymbol{E}^{r'} \boldsymbol{x}_2) \rangle] \mathbb{E}[\langle \varphi(\boldsymbol{F}^r \boldsymbol{y}_1), \varphi(\boldsymbol{F}^{r'} \boldsymbol{y}_2) \rangle]$.

A characterization of the variance of $k(\boldsymbol{z}_1, \boldsymbol{z}_2)$ in Theorem 2 is given in Appendix 4, which suggests that higher output feature dimension can reduce the variance of $k(\boldsymbol{z}_1, \boldsymbol{z}_2)$. Consequently, just by using the periodic function $\varphi(\cdot)$ and the Gaussian random projection, we can obtain infinite-order information without extra computational cost. We refer this form as "RPGaussianFull" in the experiment section. In principal, one can extract different types of high-order information by using different nonlinear functions in Eq. (4.31). Further investigations on such problems are currently beyond our scope.

In practice, when used as an intermediate layer in deep neural networks, $\sin$ and $\cos$ functions are known to be difficult and unreliable to train using back-propagation [192]. Perhaps a critical initialization step is necessary. Therefore, we perform Taylor expansion of $\sin$ and $\cos$, and only use the first term. Namely, we approximate $\sin(\boldsymbol{E}x) \approx \boldsymbol{E}\boldsymbol{x}$ and $\cos(\boldsymbol{E}x) \approx 1$. Then, we can discard the nonlinear function $\varphi(\cdot)$ in Eq. (4.31), and

[5] Specifically, $\boldsymbol{P}^r$ and $\boldsymbol{Q}^r$ are uniformly distributed on the Stiefel manifold [190, 191].

also can apply scaling factors as in Eq. (4.30). We refer this approximated version as "RPGaussian" in our experiments. Consequently, we use these components from the Taylor expansion to approximate the original bilinear form with periodic activation functions, which further approximates the feature vector in the RKHS with a compositional Gaussian kernel. In addition, we adopt a simpler version $\boldsymbol{E}^r = \frac{\sqrt{D_X}}{\sigma^r}\boldsymbol{I}_{M\times \boldsymbol{D_X}}\boldsymbol{P}^r$ and $\boldsymbol{F}^r = \frac{\sqrt{D_Y}}{\rho^r}\boldsymbol{I}_{N\times \boldsymbol{D_Y}}\boldsymbol{Q}^r$. According to [190], such a simplified version exhibits similar empirical behavior to the original version, especially when the feature dimensionality is high. Moreover, rather than regarding the Gaussian radii as hyper-parameters, we learn them via back-propagation when employing the model (4.30) as an intermediate layer in a deep neural network.

Last but not least, it is instructive to note that, in addition to what we have proposed, the distributions of $\boldsymbol{E}$ and $\boldsymbol{F}$ can be arbitrary. We can even model these distributions using deep generative models, which is the subject of our future work.

4.4 Experiment: Fine-Grained Action Parsing

Parsing fine-grained actions over time is important in many applications, which require understanding of subtle and precise operations in long-term periods, e.g. daily activities [116], surgical robots [117], human motion analysis [38] and animal behavior analysis in the lab [118]. Therefore, understanding fine-grained actions in videos has great potentials for our purpose of elderly people healthcare.

In this section, we quantitatively show the effectiveness of our bilinear pooling methods on fine-grained action parsing, based on the deep neural networks introduced in Sec. 4.3.1. As discussed above, our *statistics-based* bilinear pooling method can preserve full 2nd-order information, but has higher computational cost, while our *algebra-based* bilinear pooling method extracts approximated 2nd-order information (or higher-order) and is more lightweight. Therefore, in our experiments, we combine our statistics-based pooling method with the TCED network [59], replacing the max pooling by the bilinear pooling in each encoder. In addition, in a separate investigation, we combine our algebra-based bilinear pooling method with both TCED and MS-TCN [60]. Since

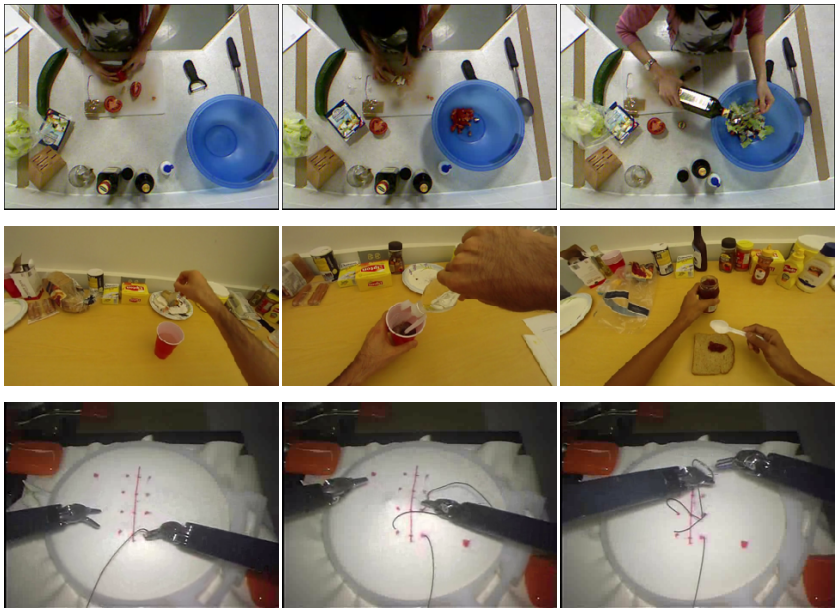

Figure 4.8: Illustration of datasets. From top to bottom: Each row presents three video frames from **50 Salads**, **GETA** and **JIGSAW**, respectively.

different methods generate different input statistics for the feature sequences, we use different training settings to achieve the best performance in each investigation, and discuss the experimental results separately.

4.4.1 Datasets

In our experiments, the input time sequence of feature vectors to the temporal network is extracted from RGB videos using a pre-trained VGG-like network [57] for the TCED network, and using a pre-trained I3D network [11] for the MS-TCN network. To perform fair comparison with other methods, we downsample the input feature sequence to achieve the same temporal resolution as in [59] for TCED and in [60] for MS-TCN, respectively. Figure 4.8 shows several examplar video frames from the three datasets introduced in the following.

50 Salads [5]. This multi-modal dataset collects 50 recordings from 25 people preparing 2 mixed salads, and each recording lasts 5-10 minutes. In our experiment, we only

use the RGB video, which has spatial resolution of 640x480 pixels and frame rate of 30 fps. The annotation is performed at two levels: (1) the *eval-level* incorporating 9 actions such as “cut”, “peel” and “add dressing”, and (2) the *mid-level* incorporating 17 fine-grained actions, some of which are derived from the high-level actions. For example, the mid-level actions “cut tomato” and “cut cucumber” are derived from the high-level action “cut”. Therefore, we can obtain two subsets from **50 Salads**, namely **50 Salads-eval** and **50 Salads-mid**. The recordings are equally split into 5 folds for cross-validation. Without direct mentioning, **50 Salads** means **50 Salads-mid**.

Georgia Tech Egocentric Activity Datasets (GTEA) [193, 116]. This dataset contains 7 daily living activities performed by 4 subjects. The videos are captured from the egocentric view at 15 fps with the resolution of 1280x720 pixels and there are 31,222 frames in the dataset. We follow the settings in [57] [59]: For each video, frame-wise labels from 11 action classes are annotated. The evaluation is based on the leave-one-subject-out scheme, namely performing cross-validation on 4-fold splits.

JHU-ISI Gesture and Skill Assessment Working Set (JIGSAWS) [194, 117]. This multi-modal dataset is collected from the *da Vinci* surgical system, which is operated by 8 surgeons with different skill levels performing 5 repetitions of 3 surgical tasks, i.e. “suturing” (39 trials), “knot-tying” (36 trials) and “needle-passing” (26 trials). Videos are captured from the endoscopic camera in a simulated setting of real surgeries. In our study we only use the videos of “suturing” since it has more trials than other tasks. The “suturing” task comprises 10 actions like “tie a knot”, “insert needle into skin” and so forth. Each video is approx. 2 minutes and contains 15 to 37 actions, which have considerably different occurrence orders from different surgeons. Similar to GTEA, in our experiments we perform evaluations in the leave-one-surgeon-out scheme.

4.4.2 Evaluation Metrics

Frame-wise accuracy. The frame-wise accuracy is defined as the correctly classified frames divided by the number of all frames. Intuitively, such measure evaluates the

accuracy from the frame-wise classification perspective. However, it ignores the temporal regularity and the action occurrence order in the label sequence.

Edit score [57]. Given the predicted sequence $P = \{p_1, p_2, ..., p_M\}$ and the ground truth sequence $Q = \{q_1, q_2, ..., q_N\}$, the edit score is defined as $(1 - d_{edit}(P, Q)) \times 100$, in which $d_{edit}(\cdot, \cdot)$ is the Levenshtein distance[6], which measures the difference between two strings. According to the definition, the edit score evaluates the temporal order of action occurrence, ignores the action temporal durations and only considers segment insertions, deletions and substitutions. Thus, such metric can provide effective evaluation of tasks, in which the action order is essential, e.g. cooking, manufacturing, surgery and so forth. However, the edit score can be negatively influenced by tiny predicted segments, and hence highly influenced by over-segmentation results.

F1 score [59]. The F1 score is defined as the harmonic mean of the precision and the recall rates of fine-grained action detection, in which the true positive detection is defined as a segment whose action label is same to the ground truth as well as the *intersection-over-union* (IoU) of the overlap with the ground truth is greater than k. In our experiment, we compute the F1 scores with IoU threshold k equal to 0.1, 0.25 and 0.5, which are denoted as F1@0.1, F1@0.25 and F1@0.5, respectively. Without directly mentioning, the F1 score means F1@0.1. According to the definition, the F1 score is an evaluation metric from the perspective of action detection, and is invariant to small temporal shifts between detection and the ground truth. However, the F1 score is negatively influenced by over-segmentation as well, since lots of tiny segments can result in a low precision rate.

4.4.3 Experiments on Our Statistics-Based Bilinear Pooling

We use the same architecture setting of the TCED in [59]: The network has a temporal encoder with the structure of "1D convolution $\rightarrow$ max pooling $\rightarrow$ 1D convolution $\rightarrow$

[6] Here is an example from Wikipedia: The Levenshtein distance from "kitten" to "sitting" is 3, since three edits are involved: (1) kitten $\rightarrow$ sitten (substitution of "s" for "k"), (2) sitten $\rightarrow$ sittin (substitution of "i" for "e") and (3) sittin $\rightarrow$ sitting (insertion of "g" at the end).

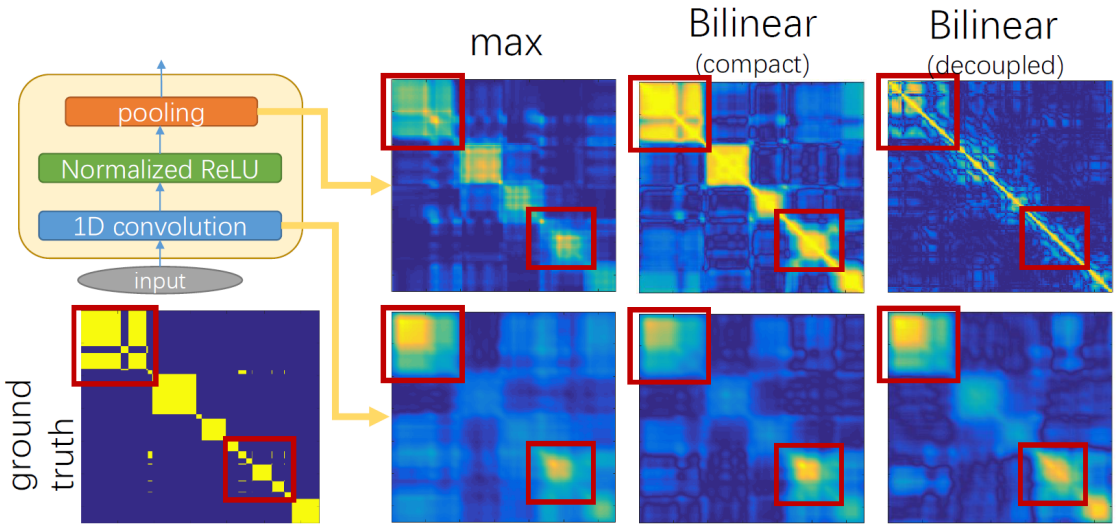

Figure 4.9: We use the features from the first encoder of TCED to show frame similarities of "rgb-01-1.avi" in **50 Salads-mid** [5]. The similarity of two frame features $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ is defined as $|\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle|$. The similarity of two (one-hot) action labels, regarded as ground truth, is computed in the same way. It is noted that in each matrix the x- and the y-axis both indicate the time range. The bilinear pooling outputs are power-and-l2 normalized. Entries in similarity matrices range between 0 (blue) and 1 (yellow). Red rectangles contain some fine-grained actions. One can see that bilinear pooling is better at recognizing fine-grained actions, but can also lead to undesired over-segmentation. Combining with the convolution layers, such over-segmentation issue vanishes. This figure was used in [6] © 2019 IEEE.

max pooling", and the associated two 1D convolution layers have 64 and 96 filters, respectively. The decoders have mirrored structures with the encoders. We replace the max pooling in the encoders by our bilinear pooling method. To train the network, for all methods we use the Adam optimizer [177] with learning rate of 0.01 or 0.001 depending on which one performs better. In addition, we set the batch size equal to 4 and stop the training after 100 epochs.

4.4.3.1 Analysis of The Bilinear Forms

We use the **50 Salads-mid** dataset to perform model analysis, because it has more fine-grained action types and longer video recordings than the other mentioned datasets.

The benefits of second-order information. Figure 4.9 illustrates the comparisons between pooling methods, where the compact bilinear pooling [69] output has the **same** dimension as the max pooling: (1) The first row in Figure 4.9 shows that bilinear pooling can capture fine-grained actions better than max pooling, whose output features tend to merge fine-grained actions into coarse-grained ones. Our proposed decoupled bilinear pooling with full second-order information performs better at recognizing fine-grained actions and suppressing off-diagonal elements. However, it can break a coarse-grained action into several segments, and can lead to undesired over-segmentation. The compact bilinear pooling outperforms max pooling on the diagonal elements, which demonstrates that the advantage of bilinear pooling is due to the second-order information rather than higher dimensionality. However, the large off-diagonal values indicate the drawback of approximating second-order information in compact bilinear pooling. (2) The second row in Figure 4.9 illustrates that bilinear pooling improves the convolution layer via backpropagation. With max pooling, many off-diagonal elements are similar to the diagonal elements, which differ from the ground truth pattern considerably. However, with bilinear pooling, the matrix patterns are more similar to the ground truth. One can also see that the output from the convolution layers does not have over-segmentation, which appears in the output of bilinear pooling.

Furthermore, we conduct a quantitative comparison on the *first split* of **50 Salads-mid**. In the format of *accuracy/edit-score/F1-score*, max pooling yields 71.03/71.8/73.09, compact bilinear pooling yields 75.41/73.75/78.96, the coupled bilinear pooling $\mathcal{B}_c(\cdot)$ yields 76.56/75.32/79.84 and the decoupled bilinear pooling $\mathcal{B}_d(\cdot)$ yields 75.11/71.06/75.79. One can see that bilinear pooling consistently outperforms max pooling. The comparison between max pooling and compact bilinear pooling also indicates the importance of the second-order information.

Comparison of different bilinear forms. Here we analyze the influence of the learnable weights in the proposed bilinear forms $\mathcal{B}_c$ and $\mathcal{B}_d$. We denote the corresponding non-learnable bilinear forms in Eq. (4.10) and Eq. (4.11) as $\mathcal{B}_c^o$ and $\mathcal{B}_d^o$, respectively. As shown in the top row of Figure 4.10, both for the coupled and decoupled bilinear forms, the one with learnable weights consistently outperforms the non-learnable counterpart,

in terms of the evaluation metrics and the robustness to the neighborhood size $|\mathcal{N}|$. This outcome is more obvious when the neighbor size is larger. This result indicates that the learnable weights, i.e. $\{\omega_\tau\}$, $\{p_\tau\}$ and $\{q_\tau\}$ in equations (4.14) and (4.17), enable the derived bilinear forms to capture more complex local temporal statistics, comparing to the standard average aggregation. Thus, in the following experiments, we only use the learnable bilinear forms. Furthermore, the decoupled bilinear form outperforms the coupled version on all the three metrics. Specifically, the decoupled bilinear form achieves *66.3/64.63/70.74* in the format of *accuracy/edit score/F1 score*, while the best performance of the coupled bilinear form is *64.73/62.15/68.89* and the baseline model (TCN$_{max}$ [59]) achieves *64.7/59.8/68.0*.

In the bottom row of Figure 4.10, we show the performance of the first-order component and the second-order component of the decoupled bilinear form. One can observe that the results derived using individual components are inferior to the results using combined bilinear forms. This fits our conjecture that first and second-order components tend to describe independent and complementary patterns in data.

Normalization and activation. Here we investigate the influence of normalization methods, which are introduced in Sec. 4.3.3, and compare different activation functions. In each individual experiment the neighbourhood size of both bilinear forms are identical. First, different normalization methods are compared in Table 4.3. One can see that l_2 normalization and l_1 normalization perform almost equally, while the normalized ReLU activation function consistently outperforms others. This result indicates that the max-normalization in intermediate layers is more suitable than others to constrain the bilinear form spectrum. Second, we show the influence of the activation functions in Table 4.4. The bilinear forms are l_2 normalized, except for the case of **NReLU**. In our experiment, training with other activation functions without l_2 normalization hardly converges, indicating the importance of constraining the spectrum of the bilinear form. Table 4.4 indicates that the **NReLU** function consistently yields superior results, suggesting that our task benefits from sparse features.

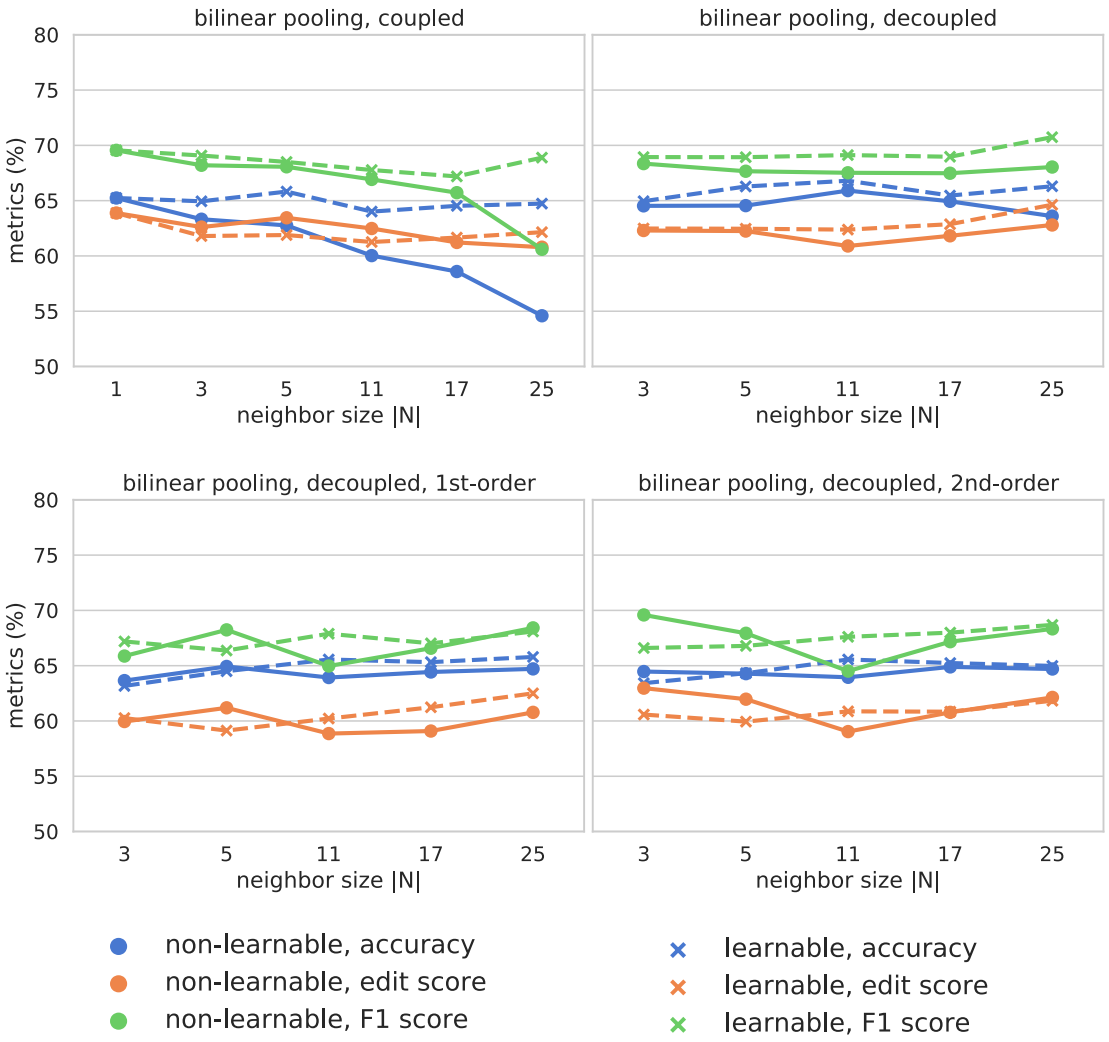

Figure 4.10: The performances w.r.t. the neighbourhood size $|\mathcal{N}|$, and the learnability of the weights. From left to right in the first row: The performance of the coupled bilinear form $\mathcal{B}_c$; the performance of the decoupled bilinear form $\mathcal{B}_d$. From left to right in the second row: The performance of the first-order component in $\mathcal{B}_d$; the performance of the second-order component in $\mathcal{B}_d$. The bilinear pooling $\mathcal{B}_d$ is referred to Eq. (4.17). This figure was used in [6] © 2019 IEEE.

4.4.3.2 Low-Dimensional Representations

One of our key contributions is to derive lower-dimensional alternatives to the explicit bilinear compositions. Comparing other dimension reduction methods, our method does

	$\mathcal{B}_c$	$\mathcal{B}_d$
NReLU	**65.82/61.89/68.5**	**66.28/62.46/68.93**
NReLU+l_2	64.92/60.01/67.33	66.09/60.02/67.38
NReLU+l_1	64.22/60.04/65.86	66.48/63.04/68.71
ReLU+l_2	64.87/59.88/66.31	64.7/61.45/68.04
ReLU+l_1	63.05/58.29/65.62	59.76/58.68/64.39

Table 4.3: Comparison of different normalization methods, in which the performances are presented in terms of *accuracy/edit score/F1 score* and the best ones for each model are highlighted in boldface. This table was used in [6] © 2019 IEEE

pooling	**ReLU** [195]	**leaky ReLU** [176]	**swish** [196]	**NReLU (Eq.** (4.8)**)** [59]	**RPN (Eq.** (4.9)**)**	**linear**
max	61.1/53.1/59.8	55.0/48.5/55.6	56.5/47.1/52.4	**63.6**/**60.4**/**64.9**	62.7/54.9/63.1	12.6/11.5/8.6
$\mathcal{B}_c$	65.5/61.1/68.4	66.4/**61.7**/**69.1**	**66.8**/59.2/67.5	64.0/61.3/67.8	64.1/56.5/64.8	63.5/48.3/55.9
$\mathcal{B}_d$	64.7/61.5/68.0	65.6/53.6/61.5	62.5/49.3/56.6	**66.8**/**62.4**/**69.1**	63.2/50.4/58.4	65.5/48.3/57.0

Table 4.4: The performances with different pooling methods and activation functions are presented in the format of *accuracy/edit score/F1 score*, in which for each model the best results are highlighted in boldface. This table was used in [6] © 2019 IEEE

not suffer from any information loss nor do we have extra computation. We compare different low-dimensional representations in the lower parts of Table 4.5, 4.6, 4.7 and 4.8. The *tensor sketch* technique [69] reduces each feature outer product from d^2 to $\frac{d(d+1)}{2}$ for fair comparison. In addition, the *LearnableProjection* [66] is implemented by a temporal convolution layer with the kernel size of 1, and the reduced dimensions are equal to ϕ_c and ϕ_d respectively for fair comparison. Note that, in our trials, other dimension reduction methods (especially the ones employing SVD) used in our local temporal pooling cause very high computational cost, lead to intractable computation, and hence are not compared. For each listed method we tested different neighborhood sizes of 5, 11 and 25, and present the best performance. Our results show that the proposed low-dimensional representations consistently outperform other dimension reduction methods. In particular, on the **50 Salads-mid** dataset, ϕ_d considerably outperforms the *LearnableProjection* counterpart, in which the accuracy is improved by 5.6%, the edit score is improved by 6.2% and the F1 score is improved by 6.1%. Here the runtime comparison between the proposed bilinear forms and their respective low-dimensional

Method	Result
Spatial CNN [57]	54.9/24.8/32.3
Spatiotemporal CNN [57]	59.4/45.9/55.9
IDT+LM [29]	48.7/45.8/44.4
Dilated TCN [59]	59.3/43.1/52.2
Bidirectional LSTM [59]	55.7/55.6/62.6
TCED$_{max}$ [59]	64.7/59.8/68.0
TCED$_{\mathcal{B}_c}$	65.8/61.9/68.5
TCED$_{\mathcal{B}_d}$	**66.3**/62.5/68.9
TCED$_{TensorSketch}$ [69]	63.4/62.6/68.5
TCED$_{\mathcal{B}_c,LearnableProjection}$	61.8/58.2/64.4
TCED$_{\mathcal{B}_d,LearnableProjection}$	60.1/56.6/62.9
TCED$_{\phi_c}$	64.7/61.3/66.8
TCED$_{\phi_d}$	65.7/**62.8**/**69.0**

Table 4.5: The comparison in **50 Salads-mid**, where the results are shown in the format of *accuracy/edit score/F1 score*. The upper part shows the comparison with other action parsing methods and the lower part shows the comparison of different dimension reduction methods. The best results are highlighted in boldface. This table was used in [6] © 2019 IEEE

representations are not performed, since the runtime is highly dependent on how the bilinear forms and the low-dimensional representations are implemented.

4.4.3.3 Comparison with The State-Of-The-Art

Table 4.5, 4.6, 4.7 and 4.8 show the performances of different methods on the datasets **50 Salads-mid**, **50 Salads-eval**, **GTEA** and **JIGSAWS**, respectively, in which TCED$_X$ denotes the temporal convolutional encoder-decoder with the pooling method X. For each method with local temporal pooling, we perform grid search on the neighborhood sets of 5, 11 and 25, and present the best one. From the tables, we can see that our proposed method can generalize well across different datasets, and produces superior or similar performances in comparison with other methods. In **50 Salads-mid**, the dataset with more fine-grained action types and longer videos than other datasets, the decoupled bilinear form, as well as its lower-dimensional representation outperform other methods

Method	Result
Spatial CNN [57]	68.0/25.5/35.0
Spatiotemporal CNN [57]	71.3/52.8/61.7
Dilated TCN [59]	71.1/46.9/55.8
Bidirectional LSTM [59]	70.9/67.7/72.2
TCED$_{max}$ [59]	73.4/**72.2**/**76.5**
TCED$_{\mathcal{B}_c}$	74.2/71.2/75.5
TCED$_{\mathcal{B}_d}$	**75.9**/71.3/76.2
TCED$_{TensorSketch}$ [69]	71.9/70.9/75.1
TCED$_{\mathcal{B}_c,LearnableProjection}$	72.0/68.8/73.4
TCED$_{\mathcal{B}_d,LearnableProjection}$	71.3/68.9/72.6
TCED$_{\phi_c}$	74.0/71.0/**76.5**
TCED$_{\phi_d}$	75.6/70.4/76.0

Table 4.6: The comparison in **50 Salads-eval**. This table was used in [6] © 2019 IEEE

Method	Result
EgoNet+TDD [53]	64.4/-/-
Spatial CNN [57]	54.8/28.7/38.3
Spatiotemporal CNN [57]	57.6/49.1/56.7
Spatiotemporal CNN+Seg [57]	52.6/53.0/57.7
Dilated TCN [59]	58.0/40.7/51.3
Bidirectional LSTM [59]	56.2/41.3/50.2
TCED$_{max}$ [59]	63.5/71.9/75.2
TCED$_{\mathcal{B}_c}$	63.6/71.7/76.4
TCED$_{\mathcal{B}_d}$	63.4/70.9/**76.8**
TCED$_{TensorSketch}$ [69]	59.8/71.2/75.2
TCED$_{\mathcal{B}_c,LearnableProjection}$	58.4/68.2/71.9
TCED$_{\mathcal{B}_d,LearnableProjection}$	58.8/70.5/74.9
TCED$_{\phi_c}$	**64.5**/71.8/75.0
TCED$_{\phi_d}$	64.4/**73.9**/76.3

Table 4.7: The comparison in **GTEA**, in which the symbol "-" denotes that the score is not available. This table was used in [6] © 2019 IEEE

Method	Result
Spatial CNN [57]	74.1/37.7/51.6
Spatiotemporal CNN [57]	77.9/67.1/77.7
Spatiotemporal CNN+Seg [57]	74.4/73.7/82.2
Dilated TCN [59]	78.0/56.8/69.7
Bidirectional LSTM [59]	74.4/73.7/82.2
TCED$_{max}$ [59]	81.2/85.6/90.3
TCED$_{\mathcal{B}_c}$	**82.6**/85.6/90.4
TCED$_{\mathcal{B}_d}$	82.2/**87.7**/**91.4**
TCED$_{TensorSketch}$ [69]	80.8/85.4/90.1
TCED$_{\mathcal{B}_c,LearnableProjection}$	79.7/82.8/88.1
TCED$_{\mathcal{B}_d,LearnableProjection}$	81.6/83.0/89.0
TCED$_{\phi_c}$	81.8/85.1/90.0
TCED$_{\phi_d}$	81.7/85.1/90.5

Table 4.8: The comparison in **JIGSAWS**. This table was used in [6] © 2019 IEEE

for all the evaluation metrics. In **50 Salads-eval**, the performance of our methods are comparable with others while with lower edit scores, probably because actions in this dataset are not sufficiently fine-grained, but our bilinear pooling produces more segments than others. Furthermore, more training epochs can increase the accuracy, yet decrease the edit score and the F1 score for our bilinear pooling models, in contrast to the max pooling baseline model. For example, after 300 epochs, TCED$_{\mathcal{B}_d}$ yields 74.7/59.2/66.7, and TCED$_{max}$ yields 63.6/71.9/75.2 for the **GTEA** dataset. A probable reason is that bilinear pooling is not good at regularizing temporal smoothness in the sequence, and has the risk of over-segmentation. Although increasing training epochs can improve recognizing actions in individual frames, the over-segmentation issue becomes more severe. Therefore, when parsing a long-term activity with both coarse-grained and fine-grained actions, one can consider to fuse the outputs from first-order pooling and second-order pooling, in order to produce accurate action understanding and avoid over-segmentation. The benefits of fusing first-order and second-order information are shown by our experiments in Sec. 4.4.4, which is based on MS-TCN and produces new state-of-the-art results.

4.4.4 Experiments on Our Algebra-Based Bilinear Pooling

We have presented experiments of combining TCED and our statistics-based bilinear pooling for fine-grained action parsing, assigning each individual frame an action label. Here, we present experiments on our proposed algebra-based bilinear pooling. In these experiments, the two input features $\boldsymbol{x}$ and $\boldsymbol{y}$ in Eq. (4.30) and Eq. (4.31) are identical, and we set the same number of rows (i.e. $M = N$) to matrices $\boldsymbol{E}^r$ and $\boldsymbol{F}^r$ for all $r = 1, \ldots, R$. Consequently, there only remain two hyper-parameters in the bilinear model, i.e. the rank R and the number of rows N of matrices. The objectives of our experiments are two-fold: (i) To verify the effectiveness of our method, we adopt the temporal convolutional encoder-decoder network (TCED) [59] due to its simple structure, and replace the max pooling in TCED by bilinear pooling as before (also as in our publication [6]). (ii) To demonstrate how our method can be used in practice, we propose a bilinear residual module to merge the first and second-order information, and combine it with the multi-stage temporal convolutional network (MS-TCN) [60] to yield state-of-the-art performance.

4.4.4.1 Action Segmentation with TCED Architecture

We use the default architecture of TCED [59], which comprises an encoder and a decoder with symmetric modules, as presented in Sec. 4.3.1. The convolutional layer in each individual encoder has 64 and 96 filters, respectively. We train the model using the Adam optimizer [177] with a fixed learning rate of 10^{-4}. Batch size is set to 8 in the mini-batch training scheme, and the training process terminates after 300 epochs. One notes that our training setting is different from the experiments on the statistics-based bilinear pooling, so the experiment results may be different. For example, the compact bilinear pooling method with the above training setting yields better performance than the results shown in Tab. 4.6 and Tab. 4.7. This indicates that the final performance could be heavily influenced by the training setting.

Table 4.9: Comparison with different bilinear pooling methods in terms of *accuracy/edit score/F1 score* and runtime (milliseconds). For each metric and each setting, the best result is in boldface. "LearnableProjection" indicates $\boldsymbol{E}$ and $\boldsymbol{F}$ in Eq. (4.30) are learned via back-propagation. "RPBinary" indicates the model Eq. (4.30) with Rademacher random projection. "RPGaussianFull" and "RPGaussian" indicate the model Eq. (4.31) with and without $\varphi(\cdot)$, respectively, in which Gaussian random projection is employed. In the column of complexity, D and d denote the input and output feature dimension of our bilinear model, respectively. One notes that $d \gg D$ in general.

		50Salads		**GTEA**	
Method	Complexity	Runtime	Performance	Runtime	Performance
Ours (RPBinary)	$\mathcal{O}(D\sqrt{d}+d)$	**68.3**	66.0/**65.9**/70.9	80.1	65.2/73.2/77.0
Ours (RPGaussian)	$\mathcal{O}(D\sqrt{d}+d)$	68.9	67.6/65.2/**72.9**	**69.9**	**66.9**/76.5/**79.8**
Ours (RPGaussianFull)	$\mathcal{O}(D\sqrt{d}+d)$	73.7	64.1/63.4/69.6	70.0	64.5/73.0/78.7
Ours (LearnableProjection)	$\mathcal{O}(D\sqrt{d}+d)$	78.6	66.4/65.0/70.5	81.6	64.8/74.0/77.5
Compact [69]	$\mathcal{O}(D+d\log d)$	95.9	67.2/65.8/71.7	120.2	65.9/75.3/78.1
Hadamard [80]	$\mathcal{O}(Dd+d)$	83.8	**67.7**/64.4/71.5	83.6	66.0/**76.6**/79.0
FBP [197]	$\mathcal{O}((2k+1)Dd+d)$	130.5	64.0/61.0/67.5	127.6	63.4/71.6/74.1

Model analysis: Comparison between bilinear forms. In this experiment, we set the rank $R = 1$ and $N = D/2$ for our methods, where D is the input feature dimension to our bilinear model. The results are shown in the first part of Table 4.9. For the performance evaluation, we repeat the experiment 3 times with different initial values of network parameters, and report the result with the highest sum of the three metrics. To evaluate the efficiency, we report the runtime [7] per batch (batch size=8) during the training phase, which is the averaged result after training with the first split for 300 epochs for each dataset. Such runtime evaluation is based on a desktop with Ubuntu 18.04, Intel Xeon(R) CPU 2.90GHz x 4, 32GB RAM, and GPU Nvidia Quadro P4000.

Overall, the results suggest that "RPGaussian" and "RPBinary" have comparable performances, and outperform "LearnableProjection" and "RPGaussianFull". While "LearnableProjection" is more flexible, it might require a more sophisticated training to achieve the same performance as our random projection methods. One notes that parameters in our proposed bilinear pooling are not updated during network training. However, the bilinear form can influence how other parameters are updated by back-propagation.

[7] Runtime measurement is implemented via a callback function in Keras, which returns the time cost of training with one mini-batch input.

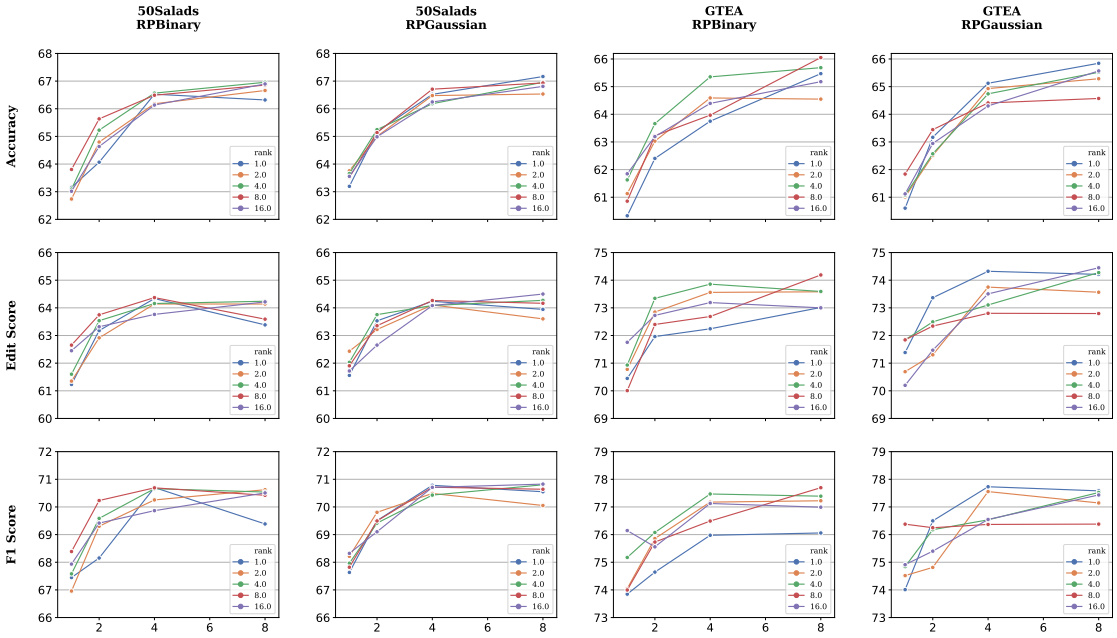

Figure 4.11: Performance of our RPBinary and RPGaussian model (see Eq. (4.30)), versus dimension / rank, on datasets **50Salads** and **GTEA**. In each plot, x-axis is the multiplier on the number of matrix rows N, y-axis is the respective performance measure, and colors denote different ranks.

For example, "RPGaussianFull" has sine and cosine as activation functions, and these periodic functions can cause the entire network hard to train. We suspect that the inferior performance of "RPGaussianFull" is due to the difficulty of training.

In addition, we compare our methods with three widely used bilinear pooling methods, i.e., **Compact** [69], **Hadamard** [80] and **FBP** [197]. We use the same output feature dimension for fair comparison. The results are shown in the second part of Table 4.9. They suggest that these three methods perform inferior or comparably in terms of the three metrics of action parsing, and are clearly less efficient, than our methods.

Model analysis: Investigation on the hyper-parameters of our bilinear forms. Here we investigate the influence of the rank and the output feature dimension of RPBinary and RPGaussian. One can recall that RPBinary and RPGaussian share the same bilinear model Eq. (4.30), but with matrix entries sampled from different distributions. Specifically, RPBinary has matrix entries sampled from Rademacher distribution, and

RPGaussian has matrix entries sampled from Gaussian distribution with orthogonality constraints. Figure 4.11 shows the dependence of the model performance on ranks $R \in \{1, 2, 4, 8, 16\}$, and matrix rows $N \in \{1, 2, 4, 8\} \times [\sqrt{D}]$ where $[n]$ denotes the nearest integer to n. In this case, the output feature dimension is then $\{1, 4, 16, 64\} \times D$. In all plots, the performance increases consistently with the matrix row N (hence the output feature dimension). This result is in line with our theoretical analysis on the kernel approximation error bound (see Corollary 1 and Appendix 4): Larger values of M and N can yield lower variance upper bounds, hence better kernel approximation. One can also observe that the performance saturates when further increasing the output feature dimension. This is due to the limited capacity of the corresponding RKHS. Moreover, one can observe that increasing the rank may not consistently yield better performance. An optimal rank depends on the dataset and the applied bilinear model. In practice, one may choose a value to balance the complexity and the representativeness. Or, one can use validation dataset to choose an optimal rank.

4.4.4.2 Action Segmentation with MS-TCN

In this section, we demonstrate how to effectively incorporate our method into the state-of-the-art action parsing network, MS-TCN [60], to superior performance.

Implementation Details. Based on our previous experimental results, we set rank $R = 4$ and $N = D/2$ for our bilinear model. Furthermore, we propose a bilinear residual module to merge the first and the second-order information, as illustrated in Figure 4.12. First, we use bilinear pooling to extract the second-order information, and then use a regularized power normalization [6] to densify the features, and use channel-wise max-normalization to re-scale the feature values [59]. Afterwards, we use a convolution layer to reduce the feature dimension to the number of action classes. Since the second-order information tends to partition an action into smaller segments (see Fig. 4.9 or [6, Figure 1]), we use a large convolution receptive field 25 according to [59]. To prevent overfitting, we use a dropout layer in the bilinear residual module, and then we compute the average

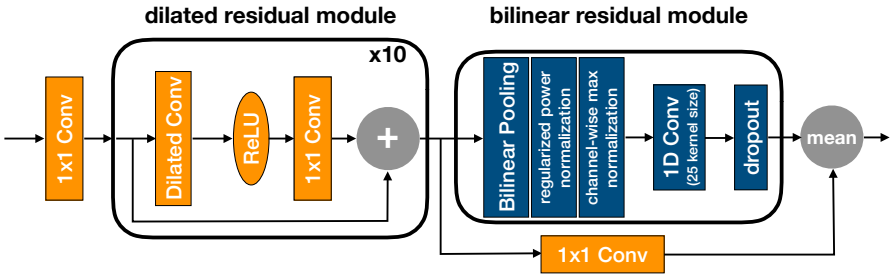

Figure 4.12: A combination of MS-TCN [60] and our bilinear pooling method, in which the blue layers are proposed by us, and the orange layers are proposed by MS-TCN.

between the first-order information and the second-order information. One can see Figure 4.12, in which the layers demonstrated above are denoted by blue.

Our bilinear residual module is applied at the end of each single stage of MS-TCN. To conduce fair comparison with the baseline MS-TCN model, we keep other model configurations and the loss function (including the hyper-parameters) unchanged. Similar to [60], we use the Adam [177] optimizer with a learning rate of 0.0005. The mini-batch size is set to 1 during training. Since the dataset **50Salads** has more training samples than **GTEA**, we set the number of training epochs to 50 for **GTEA** and 70 for **50Salads**, respectively. Since more training epochs have a higher risk of overfitting, we set the dropout ratio to 0.5 for **GTEA**, and 0.7 for **50Salads**.

Result. We compare our method with several state-of-the-art methods on the action segmentation task. As shown in Table 4.10, our models (especially RPBinary and RPGaussian) consistently outperform the state-of-the-art methods (TCN, TDRN, MS-TCN), validating the effectiveness of the proposed bilinear model. Note that, even with RPGaussianFull, which is hard to train by back-propagation, our bilinear form achieves competitive performance on the GTEA dataset. For a fair and complete comparison, we further integrate three widely used light-weight bilinear models into the MS-TCN model, namely, **Compact** [69], **Hadamard** [80] and **FBP** [197]. Again, the proposed bilinear models show superior performance on most of the evaluation metrics. Together with the results presented in Table 4.9, they clearly demonstrate the representational power and the computational efficiency of the proposed bilinear models. Figure 4.13

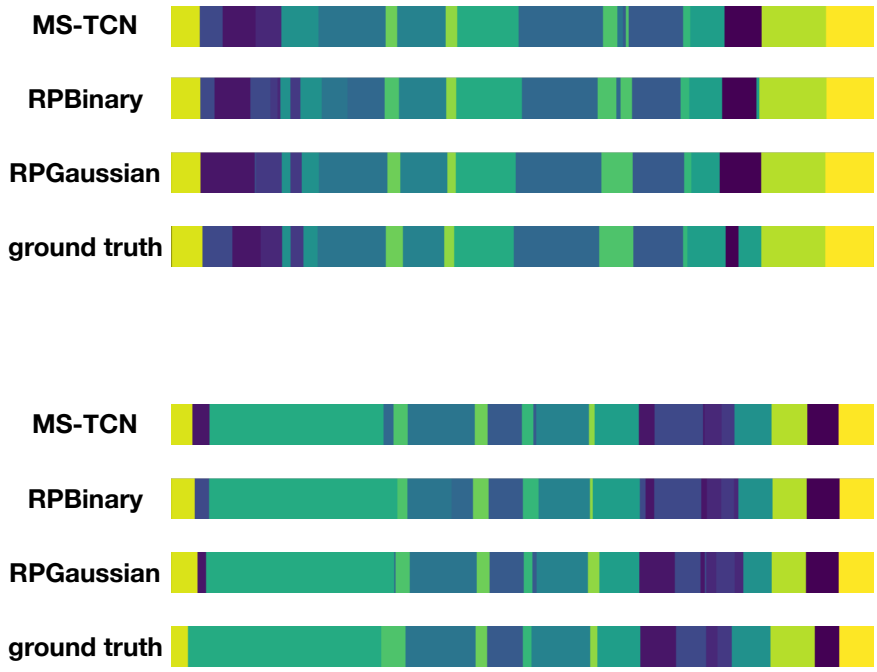

Figure 4.13: Qualitative comparison with different methods on two videos in the **50Salads** datasets. The action labels are coded by the color map *viridis*.

shows two qualitative segmentation results with different methods. One can find that our bilinear pooling methods outperform the ms-tcn method [60] w.r.t. find-grained action recognition.

Table 4.10: Comparison with other models on temporal action segmentation task. The best results are in **boldface**, the second-best results are underlined.

	50 Salads					GTEA				
	Acc.	Edit	F1@0.1	F1@0.25	F1@0.5	Acc.	Edit	F1@0.1	F1@0.25	F1@0.5
Ours (RPBinary)	79.9	70.7	78.0	75.2	65.4	77.0	81.4	86.5	**84.5**	71.7
Ours (RPGaussian)	80.6	**71.0**	**78.4**	**75.8**	66.7	**77.2**	82.2	86.7	84.3	**72.7**
Ours (RPGaussianFull)	77.4	67.5	74.9	70.9	60.7	76.8	**82.5**	**87.3**	84.3	71.8
TCN [59]	64.7	59.8	68.0	63.9	52.6	64.0	-	72.2	69.3	56.0
TDRN [61]	68.1	66.0	72.9	68.5	57.2	70.1	74.1	79.2	74.4	62.7
MS-TCN [60]	80.7	67.9	76.3	74.0	64.5	76.3	79.0	85.8	83.4	69.8
MS-TCN + Compact [69]	**81.1**	70.7	77.6	75.1	**67.1**	75.8	80.9	86.0	83.7	70.2
MS-TCN + Hadamard [80]	78.3	68.3	75.3	72.4	62.5	77.0	81.5	86.0	83.5	70.7
MS-TCN + FBP [197]	78.5	70.0	76.5	73.6	64.6	75.4	79.3	84.0	81.8	69.9

4.5 Discussions on Deep Learning-Based Human Action Parsing

In this chapter, we present how to use a deep neural network to perform human action parsing in an end-to-end manner, and fall understanding in videos. In addition, we use bilinear pooling to extract second-order information (or higher-order) for fine-grained action parsing. Compared to designing a sophisticated neural network, e.g. [61, 60], our bilinear pooling methods are well interpreted by RKHS theories, and is employed as a generic neural network layer. When combining with a deep neural network like MS-TCN [60], the performance for fine-grained action parsing is further improved.

We investigate bilinear pooling from two perspectives: The first perspective is on **statistics**, especially on covariance computation. Conventionally, average pooling or max pooling is normally used in a deep neural net, which only extracts first-order information in terms of mean or max. We consider that incorporating second-order statistics is able to differentiate two actions with identical first-order features but different second-order features. To make the temporal local statistics data-adaptive, we replace the conventional averaging operation for computing covariance by weighted sum with learnable weights. The second perspective is on **multilinear algebra**. Specifically, we expect to extract second-order information via three-way tensor product (see. Eq. (4.24)). In contrast to computing the covariance to obtain 2nd-order information, whether the output of such tensor product represents second-order information or not depends on the tensor entries.

We prove that when the tensor entries are sampled from Rademacher distribution, the tensor product can approximate the feature map to a RKHS with a polynomial kernel and hence can produce approximated second-order information. Also we prove that then the tensor entries are sampled from Gaussian distribution, the tensor product can approximate the feature map to a RKHS with a Gaussian kernel and hence can produce approximated infinite-order information.

A common issue of second-order information is the high dimensionality. To overcome such issue, we investigate how to reduce the dimensionality while preserving the representativeness based on RKHS theories: For our statistics-based bilinear pooling, we use half of entries in the covariance matrix (or SSCP matrix), and a scalar factor $\sqrt{2}$ to preserve the full 2nd-order information. For our algebra-based bilinear pooling, we use random features to approximate reproducing kernels, and hence approximate 2nd-order or even infinite-order information.

Like other applications such as fine-grained image segmentation [62], fine-grained image classification [65] and action recognition based on Fisher vectors [122], second-order information via bilinear pooling is also beneficial for our fine-grained action parsing task, which is shown in Figure 4.9 and in our experimental results. In practice, one can consider the following aspects to maximize the benefits of using bilinear pooling: (1) A second-order feature vector is in general of higher dimension than a first-order feature vector, otherwise the second-order information cannot be effectively represented. For scenarios without high efficiency requirements, higher dimensionality is preferred. (2) When using bilinear pooling as an intermediate layer, it is important to apply normalization after bilinear pooling, in order to constrain the bilinear form spectra, which can stabilize the training process and ensure effective performance. (3) A high-dimensional second-order feature can heavily lead to overfitting. Therefore, extra regularization tricks, e.g. dropout, should be applied. As verified by our experiments, the dropout ratio can be very high. (4) To obtain a better performance, first-order information and second-order information are better to be combined.

One notes that our bilinear models work as intermediate layers in a deep neural network, and hence they can be used in arbitrary network architectures to extract high-order

information. Therefore, the bilinear models proposed in Sec. 4.3 can also be used in the methods in Sec. 4.1 and Sec. 4.2 to realize fine-grained action understanding in videos.

5Conclusion and Outlooks

In this thesis, we study human action parsing in the untrimmed video, which contains a continuous and long-term human activity incorporating different consecutive actions. The task of human action parsing is to perform frame-wise action understanding, i.e., assigning each individual frame in the input video an action label/cluster ID. Since our studies in this thesis aim at providing effective solutions to applications of elderly people healthcare, all our experiments in this thesis focus on the scenario of *single person*, *indoor environment*, and activities with *various time durations and granularity levels*, which occur in common daily lives of humans.

We conduct researches on both unsupervised methods and supervised methods for human action parsing in this thesis. Specifically, the contribution of this thesis is summarized as follows:

- **Hierarchical dynamic clustering.** Inspired by the conventional *bag-of-visual-words* method, we propose the fully unsupervised hierarchical dynamic clustering (HDC) framework, in order to determine temporal segment boundaries and assign each temporal segment a cluster ID, which is equivalent to assigning each frame in the input sequence a cluster ID. Our HDC framework has two dynamic clustering modules for codebook learning and temporal segment clustering, respectively, and a local temporal pooling module to produce segment boundaries and action patterns from individual segments. For application, we show how to use the proposed HDC for temporal action segmentation in videos (or motion capture recordings), and unsupervised fainting detection in omni-directional videos. As verified by experiments, the proposed HDC method outperforms other relevant unsupervised action parsing methods in terms of accuracy and efficiency. Details are demonstrated in Chapter 3.

- **A novel dynamic clustering algorithm.** We propose a novel dynamic clustering algorithm to group individual elements in a time sequence of generic feature vectors. Compared to conventional clustering methods, such as k-means and

spectral clustering, the proposed dynamic clustering method produces better grouping results and runs faster. Details of advantages, method analysis, and comparisons with other methods are demonstrated in Sec. 3.2.

- **Local temporal pooling.** The local temporal pooling scheme is to determine temporal boundaries of segments, and extract action patterns from individual temporal segments. To overcome the "chicken-and-egg" problem between determining accurate temporal boundaries and extracting representative patterns from temporal segments, we propose a kernelized cut-based pooling scheme, which iteratively performs temporal boundary localization and feature aggregation, and a motion-energy based pooling scheme, which performs temporal boundary localization and feature aggregation from different information sources. The advantages of our proposed methods over the conventional sliding window-based pooling are demonstrated in Sec. 3.4.

- **End-to-end supervised human action parsing.** We investigate spatiotemporal convolution neural networks to conduct supervised human action parsing in untrimmed videos, aiming to assign each individual frame an action label. In particular, we investigate bilinear pooling from two perspectives, and use it to extract high-order information for fine-grained action parsing. Details are demonstrated in Chapter 4.

- **Explanation of fall understanding in videos.** We perform extensive empirical studies on how a spatiotemporal convolution encoder-decoder network understands falls from videos. Although many studies have proposed effective solutions for fall detection, to our knowledge, we are the first to investigate the underlying reasons. Details of our empirical studies are demonstrated in Sec. 4.2.

- **Local temporal bilinear pooling.** Despite effectiveness of temporal encoder-decoder network for action understanding, high-order information ($\geq$ second-order) is ignored, which is essential for fine-grained understanding tasks. We propose two novel bilinear pooling methods, i.e. statistics-based bilinear pooling, which preserves full second-order information with a lossless dimension reduction method, and algebra-based bilinear pooling, which approximates second-order and

even infinite-order information with a low computational cost. Our bilinear pooling methods are generic, can be regarded as intermediate network layers, and hence can be employed in arbitrary network architectures. The experimental results show that our proposed methods outperform other state-of-the-art methods on various fine-grained action parsing datasets. Details of theories and experiments are demonstrated in Sec. 4.3 and Sec. 4.4.

Based on our studies on human action parsing in this thesis, we have some outlooks on related study topics in future, which could be "low-hanging fruits":

- **Online action understanding and prediction.** One can recall that the convolution used in our temporal encoder-decoder network in Chapter 4 is non-causal. Despite its effectiveness on action understanding, non-causal temporal convolution is not suitable for action prediction/understanding in an online fashion. Therefore, an interesting topic is to predict fine-grained actions in future frames, based on the combination of our bilinear pooling methods and a causal temporal convolution network, such as Wavenet [198], or a recurrent neural network, such as LSTM [199] [153].
- **Human action generation.** In this thesis, we focus on action parsing, converting video frames into a sequence of action labels. It is interesting to consider its inverse problem: Given a sequence of action labels, how to generate natural human actions, which are visualized in terms of videos or body skeletons. One can note that the employed temporal convolution encoder-decoder network is symmetric, and hence we conjecture that we could flip the encoder-decoder architecture, and combine it with generative models for action generation.
- **Multi-modal input.** In this thesis, we mainly focus on video frame analysis, and also use motion capture data as input for experiments in Sec. 3.6.1. To improve action understanding, one can use different kinds of hardware to capture multi-modal input data. For example, the event-based camera [200] is able to provide much higher temporal resolution (up to microseconds) than the conventional RGB camera, and applying such high temporal resolution events has great potentials for high-accurate action parsing. The temporal boundaries are located up to the

microsecond precision level, and the perceiving system for example can response falls of elderly people in a much faster manner.

Definitely, future studies are not limited by the listed topics above. We expect that our studies in this thesis can lead to effective solutions for elderly people healthcare in practice, as well as other applications. Also, we expect that our studies in this thesis can inspire other researchers for their own studies.

Bibliography

[1] Mba, C.J.: Population ageing in ghana: research gaps and the way forward. Journal of aging research **2010** (2010)

[2] Hassan, M., Choutas, V., Tzionas, D., Black, M.J.: Resolving 3D human pose ambiguities with 3D scene constraints. In: International Conference on Computer Vision. (2019) 2282–2292

[3] Rohrbach, M., Amin, S., Andriluka, M., Schiele, B.: A database for fine grained activity detection of cooking activities. In: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, IEEE (2012) 1194–1201

[4] Schuldt, C., Laptev, I., Caputo, B.: Recognizing human actions: A local svm approach. In: Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 3 - Volume 03. Volume 3 of ICPR '04., Washington, DC, USA, IEEE, IEEE Computer Society (2004) 32–36

[5] Stein, S., McKenna, S.J.: Combining embedded accelerometers with computer vision for recognizing food preparation activities. In: Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing, ACM (2013) 729–738

[6] Zhang, Y., Tang, S., Muandet, K., Jarvers, C., Neumann, H.: Local temporal bilinear pooling for fine-grained action parsing. In: Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR). (2019)

[7] Tang, S., Andriluka, M., Milan, A., Schindler, K., Roth, S., Schiele, B.: Learning people detectors for tracking in crowded scenes. In: Proceedings of the IEEE international conference on computer vision. (2013) 1049–1056

[8] Cao, Z., Simon, T., Wei, S.E., Sheikh, Y.: Realtime multi-person 2d pose estimation using part affinity fields. arXiv preprint arXiv:1611.08050 (2016)

[9] Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., Fei-Fei, L.: Large-scale video classification with convolutional neural networks. In: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition. (2014) 1725–1732

[10] Oneata, D., Verbeek, J., Schmid, C.: Action and event recognition with fisher vectors on a compact feature set. In: 2013 IEEE International Conference on Computer Vision. (2013) 1817–1824

[11] Carreira, J., Zisserman, A.: Quo vadis, action recognition? a new model and the kinetics dataset. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE (2017) 4724–4733

[12] Peng, X., Wang, L., Wang, X., Qiao, Y.: Bag of visual words and fusion methods for action recognition: Comprehensive study and good practice. Computer Vision and Image Understanding **150** (2016) 109–125

[13] Aggarwal, J., Ryoo, M.: Human activity analysis: A review. ACM Comput. Surv. **43** (2011) 16:1–16:43

[14] Chaaraoui, A.A., Climent-Pérez, P., Flórez-Revuelta, F.: A review on vision techniques applied to human behaviour analysis for ambient-assisted living. Expert Systems with Applications **39** (2012) 10873–10888

[15] Rashidi, P., Mihailidis, A.: A survey on ambient-assisted living tools for older adults. IEEE journal of biomedical and health informatics **17** (2013) 579–590

[16] Sathyanarayana, S., Satzoda, R.K., Sathyanarayana, S., Thambipillai, S.: Vision-based patient monitoring: a comprehensive review of algorithms and technologies. Journal of Ambient Intelligence and Humanized Computing **9** (2018) 225–251

[17] Mathie, M.J., Coster, A.C., Lovell, N.H., Celler, B.G.: Accelerometry: providing an integrated, practical method for long-term, ambulatory monitoring of human movement. Physiological measurement **25** (2004) R1

[18] Mainwaring, A., Culler, D., Polastre, J., Szewczyk, R., Anderson, J.: Wireless sensor networks for habitat monitoring. In: Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications, Acm (2002) 88–97

[19] Lowe, S.A., ÓLaighin, G.: Monitoring human health behaviour in one's living environment: a technological review. Medical engineering & physics **36** (2014) 147–168

[20] Eisa, S., Moreira, A.: A behaviour monitoring system (bms) for ambient assisted living. Sensors **17** (2017) 1946

[21] Shen, H., Yu, S.I., Yang, Y., Meng, D., Hauptmann, A.: Unsupervised video adaptation for parsing human motion. In: European Conference on Computer Vision, Springer (2014) 347–360

[22] Donahue, J., Anne Hendricks, L., Guadarrama, S., Rohrbach, M., Venugopalan, S., Saenko, K., Darrell, T.: Long-term recurrent convolutional networks for visual recognition and description. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2015) 2625–2634

[23] Zhang, Y., Layher, G., Neumann, H.: Continuous activity understanding based on accumulative pose-context visual patterns. In: Seventh International Conference on Image Processing Theory, Tools and Applications, IEEE (2017) 1–6

[24] Cheng, Y., Fan, Q., Pankanti, S., Choudhary, A.: Temporal sequence modeling for video event detection. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2014) 2227–2234

[25] Wang, L., Xiong, Y., Lin, D., Van Gool, L.: Untrimmednets for weakly supervised action recognition and detection. In: Proc. CVPR. (2017)

[26] Zhao, Y., Xiong, Y., Wang, L., Wu, Z., Tang, X., Lin, D.: Temporal action detection with structured segment networks. arXiv preprint arXiv:1704.06228 (2017)

[27] Shou, Z., Wang, D., Chang, S.F.: Temporal action localization in untrimmed videos via multi-stage cnns. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2016) 1049–1058

[28] Buch, S., Escorcia, V., Shen, C., Ghanem, B., Niebles, J.C.: Sst: Single-stream temporal action proposals. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE (2017) 6373–6382

[29] Richard, A., Gall, J.: Temporal action detection using a statistical language model. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2016) 3131–3140

[30] Escorcia, V., Heilbron, F.C., Niebles, J.C., Ghanem, B.: Daps: Deep action proposals for action understanding. In: European Conference on Computer Vision, Springer (2016) 768–784

[31] Zhou, F., De la Torre, F., Hodgins, J.K.: Hierarchical aligned cluster analysis for temporal clustering of human motion. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) **35** (2013) 582–596

[32] Krüger, B., Vögele, A., Willig, T., Yao, A., Klein, R., Weber, A.: Efficient unsupervised temporal segmentation of motion data. IEEE Transactions on Multimedia **19** (2017) 797–812

[33] Li, S., Li, K., Fu, Y.: Temporal subspace clustering for human motion segmentation. In: IEEE International Conference on Computer Vision (ICCV). (2015) 4453–4461

[34] Verbeek, J.J., Vlassis, N., Kröse, B.: Efficient greedy learning of gaussian mixture models. Neural computation **15** (2003) 469–485

[35] Blei, D.M., Jordan, M.I.: Variational inference for dirichlet process mixtures. Bayesian analysis **1** (2006) 121–143

[36] Gershman, S.J., Blei, D.M.: A tutorial on bayesian nonparametric models. Journal of Mathematical Psychology **56** (2012) 1–12

[37] Hughes, M.C., Sudderth, E.: Memoized online variational inference for dirichlet process mixture models. In: Advances in Neural Information Processing Systems. (2013) 1133–1141

[38] Zhang, Y., Sun, H., Tang, S., Neumann, H.: Temporal human action segmentation via dynamic clustering. arXiv preprint arXiv:1803.05790 (2018)

[39] Zhang, Y., Tang, S., Sun, H., Neumann, H.: Human motion parsing by hierarchical dynamic clustering. In: Proceedings of the British Machine Vision Conference (BMVC), BMVA Press (2018) 269

[40] Perry, J., Davids, J.R., et al.: Gait analysis: normal and pathological function. Journal of Pediatric Orthopaedics **12** (1992) 815

[41] Nieto-Hidalgo, M., Ferrández-Pastor, F.J., Valdivieso-Sarabia, R.J., Mora-Pascual, J., García-Chamizo, J.M.: Gait analysis using computer vision based on cloud platform and mobile device. Mobile Information Systems **2018** (2018)

[42] Muro-De-La-Herran, A., Garcia-Zapirain, B., Mendez-Zorrilla, A.: Gait analysis methods: An overview of wearable and non-wearable systems, highlighting clinical applications. Sensors **14** (2014) 3362–3394

[43] Spildooren, J., Vinken, C., Van Baekel, L., Nieuwboer, A.: Turning problems and freezing of gait in parkinson's disease: a systematic review and meta-analysis. Disability and rehabilitation (2018) 1–11

[44] Mannini, A., Sabatini, A.M.: A hidden markov model-based technique for gait segmentation using a foot-mounted gyroscope. In: Engineering in Medicine and Biology Society, EMBC, 2011 Annual International Conference of the IEEE, IEEE (2011) 4369–4373

[45] Fontecha, J., Navarro, F.J., Hervás, R., Bravo, J.: Elderly frailty detection by using accelerometer-enabled smartphones and clinical information records. Personal and ubiquitous computing **17** (2013) 1073–1083

[46] Wang, L., Tan, T., Ning, H., Hu, W.: Silhouette analysis-based gait recognition for human identification. IEEE transactions on pattern analysis and machine intelligence **25** (2003) 1505–1518

[47] Popoola, O.P., Wang, K.: Video-based abnormal human behavior recognition—a review. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) **42** (2012) 865–878

[48] Javan Roshtkhari, M., Levine, M.D.: Online dominant and anomalous behavior detection in videos. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2013) 2611–2618

[49] Nater, F., Grabner, H., Van Gool, L.: Exploiting simple hierarchies for unsupervised human behavior analysis. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE (2010) 2014–2021

[50] Mehran, R., Oyama, A., Shah, M.: Abnormal crowd behavior detection using social force model. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2009) 935–942

[51] Zhang, Y., Tang, S., Sun, H., Neumann, H.: Human motion parsing by hierarchical dynamic clustering. In: BMVC. (2018)

[52] Fathi, A., Rehg, J.M.: Modeling actions through state changes. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2013) 2579–2586

[53] Singh, S., Arora, C., Jawahar, C.: First person action recognition using deep learned descriptors. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2016) 2620–2628

[54] Wang, L., Qiao, Y., Tang, X.: Action recognition with trajectory-pooled deep-convolutional descriptors. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2015) 4305–4314

[55] Singh, B., Marks, T.K., Jones, M., Tuzel, O., Shao, M.: A multi-stream bi-directional recurrent neural network for fine-grained action detection. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE (2016) 1961–1970

[56] Lea, C., Vidal, R., Hager, G.D.: Learning convolutional action primitives for fine-grained action recognition. In: Robotics and Automation (ICRA), 2016 IEEE International Conference on, IEEE (2016) 1642–1649

[57] Lea, C., Reiter, A., Vidal, R., Hager, G.D.: Segmental spatiotemporal cnns for fine-grained action segmentation. In: European Conference on Computer Vision, Springer (2016) 36–52

[58] Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)

[59] Lea, C., Flynn, M.D., Vidal, R., Reiter, A., Hager, G.D.: Temporal convolutional networks for action segmentation and detection. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2017) 1003–1012

[60] Farha, Y.A., Gall, J.: Ms-tcn: Multi-stage temporal convolutional network for action segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2019) 3575–3584

[61] Lei, P., Todorovic, S.: Temporal deformable residual networks for action segmentation in videos. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2018) 6742–6751

[62] Carreira, J., Caseiro, R., Batista, J., Sminchisescu, C.: Semantic segmentation with second-order pooling. In: European Conference on Computer Vision, Springer (2012) 430–443

[63] Moghimi, M., Belongie, S.J., Saberian, M.J., Yang, J., Vasconcelos, N., Li, L.J.: Boosted convolutional neural networks. In: BMVC. (2016)

[64] Li, P., Xie, J., Wang, Q., Zuo, W.: Is second-order information helpful for large-scale visual recognition. In: IEEE international conference on computer vision (ICCV). IEEE. (2017) 2070–2078

[65] Lin, T.Y., RoyChowdhury, A., Maji, S.: Bilinear cnn models for fine-grained visual recognition. In: Proceedings of the IEEE International Conference on Computer Vision. (2015) 1449–1457

[66] Lin, T.Y., RoyChowdhury, A., Maji, S.: Bilinear convolutional neural networks for fine-grained visual recognition. IEEE transactions on pattern analysis and machine intelligence **40** (2018) 1309–1322

[67] Lin, T.Y., Maji, S.: Improved bilinear pooling with cnns. arXiv preprint arXiv:1707.06772 (2017)

[68] Lin, T.Y., Maji, S., Koniusz, P.: Second-order democratic aggregation. In: ECCV. (2018) 639–656

[69] Gao, Y., Beijbom, O., Zhang, N., Darrell, T.: Compact bilinear pooling. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2016) 317–326

[70] Suh, Y., Wang, J., Tang, S., Mei, T., Lee, K.M.: Part-aligned bilinear representations for person re-identification. In: European Conference on Computer Vision (ECCV). (2018)

[71] Kong, S., Fowlkes, C.: Low-rank bilinear pooling for fine-grained classification. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE (2017) 7025–7034

[72] Li, P., Xie, J., Wang, Q., Gao, Z.: Towards faster training of global covariance pooling networks by iterative matrix square root normalization. arXiv preprint arXiv:1712.01034 (2017)

[73] Wei, X., Zhang, Y., Gong, Y., Zhang, J., Zheng, N.: Grassmann pooling as compact homogeneous bilinear pooling for fine-grained visual classification. In: Proceedings of the European Conference on Computer Vision (ECCV). (2018) 355–370

[74] Yu, C., Zhao, X., Zheng, Q., Zhang, P., You, X.: Hierarchical bilinear pooling for fine-grained visual recognition. In: European Conference on Computer Vision, Springer (2018) 595–610

[75] Yu, K., Salzmann, M.: Statistically-motivated second-order pooling. In: The European Conference on Computer Vision (ECCV). (2018)

[76] Wang, Q., Li, P., Zhang, L.: G2denet: Global gaussian distribution embedding network and its application to visual recognition. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Volume 1. (2017) 3

[77] Gou, M., Xiong, F., Camps, O., Sznaier, M.: Monet: Moments embedding network. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2018) 3175–3183

[78] Simon, M., Gao, Y., Darrell, T., Denzler, J., Rodner, E.: Generalized orderless pooling performs implicit salient matching. In: International Conference on Computer Vision (ICCV). (2017)

[79] Fukui, A., Park, D.H., Yang, D., Rohrbach, A., Darrell, T., Rohrbach, M.: Multimodal compact bilinear pooling for visual question answering and visual grounding. arXiv preprint arXiv:1606.01847 (2016)

[80] Kim, J.H., On, K.W., Lim, W., Kim, J., Ha, J.W., Zhang, B.T.: Hadamard product for low-rank bilinear pooling. arXiv preprint arXiv:1610.04325 (2016)

[81] Yu, Z., Yu, J., Fan, J., Tao, D.: Multi-modal factorized bilinear pooling with co-attention learning for visual question answering. In: Proc. IEEE Int. Conf. Comp. Vis. Volume 3. (2017)

[82] Tenenbaum, J.B., Freeman, W.T.: Separating style and content with bilinear models. Neural computation **12** (2000) 1247–1283

[83] Diba, A., Sharma, V., Van Gool, L.: Deep temporal linear encoding networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Volume 1. (2017)

[84] Feichtenhofer, C., Pinz, A., Zisserman, A.: Convolutional two-stream network fusion for video action recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2016) 1933–1941

[85] Hu, J.F., Zheng, W.S., Pan, J., Lai, J., Zhang, J.: Deep bilinear learning for rgb-d action recognition. In: Proceedings of the European Conference on Computer Vision (ECCV). (2018) 335–351

[86] Yue, K., Sun, M., Yuan, Y., Zhou, F., Ding, E., Xu, F.: Compact generalized non-local network. arXiv preprint arXiv:1810.13125 (2018)

[87] Wang, X., Girshick, R., Gupta, A., He, K.: Non-local neural networks. arXiv preprint arXiv:1711.07971 **10** (2017)

[88] Girdhar, R., Ramanan, D.: Attentional pooling for action recognition. In: Advances in Neural Information Processing Systems. (2017) 34–45

[89] Pham, N., Pagh, R.: Fast and scalable polynomial kernels via explicit feature maps. In: Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM (2013) 239–247

[90] Koniusz, P., Yan, F., Gosselin, P.H., Mikolajczyk, K.: Higher-order occurrence pooling for bags-of-words: Visual concept detection. IEEE transactions on pattern analysis and machine intelligence **39** (2017) 313–326

[91] Cui, Y., Zhou, F., Wang, J., Liu, X., Lin, Y., Belongie, S.J.: Kernel pooling for convolutional neural networks. In: CVPR. Volume 1. (2017) 7

[92] Zhang, Y., Tang, S., Muandet, K., Jarvers, C., Neumann, H.: Local temporal bilinear pooling for fine-grained action parsing. arXiv preprint arXiv:1812.01922 (2018)

[93] Zhang, Y., Muandet, K., Ma, Q., Neumann, H., Tang, S.: Low-rank random tensor for bilinear pooling. arXiv preprint arXiv:1906.01004 (2019)

[94] Igual, R., Medrano, C., Plaza, I.: Challenges, issues and trends in fall detection systems. Biomedical engineering online **12** (2013) 66

[95] Mubashir, M., Shao, L., Seed, L.: A survey on fall detection: Principles and approaches. Neurocomputing **100** (2013) 144–152

[96] Rougier, C., Meunier, J., St-Arnaud, A., Rousseau, J.: Robust video surveillance for fall detection based on human shape deformation. IEEE Transactions on circuits and systems for video Technology **21** (2011) 611–622

[97] Vishwakarma, V., Mandal, C., Sural, S.: Automatic detection of human fall in video. In: International conference on pattern recognition and machine intelligence, Springer (2007) 616–623

[98] Piccardi, M.: Background subtraction techniques: a review. In: IEEE international conference on Systems, man and cybernetics. Volume 4., IEEE (2004) 3099–3104

[99] Anderson, D., Keller, J.M., Skubic, M., Chen, X., He, Z.: Recognizing falls from silhouettes. In: Proceedings of the 28th IEEE EMBS Annual International Conference, IEEE (2006) 6388–6391

[100] Töreyin, B.U., Dedeoğlu, Y., Çetin, A.E.: Hmm based falling person detection using both audio and video. In: International Workshop on Human-Computer Interaction, Springer (2005) 211–220

[101] Stone, E.E., Skubic, M.: Fall detection in homes of older adults using the microsoft kinect. IEEE journal of biomedical and health informatics **19** (2015) 290–301

[102] Solbach, M.D., Tsotsos, J.K.: Vision-based fallen person detection for the elderly. arXiv preprint arXiv:1707.07608 (2017)

[103] Simonyan, K., Zisserman, A.: Two-stream convolutional networks for action recognition in videos. In: Advances in neural information processing systems (NIPS). (2014) 568–576

[104] Carreira, J., Zisserman, A.: Quo vadis, action recognition? a new model and the kinetics dataset. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2017)

[105] Yeung, S., Russakovsky, O., Mori, G., Fei-Fei, L.: End-to-end learning of action detection from frame glimpses in videos. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2016) 2678–2687

[106] Gkioxari, G., Malik, J.: Finding action tubes. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE (2015) 759–768

[107] Shou, Z., Chan, J., Zareian, A., Miyazawa, K., Chang, S.F.: Cdc: convolutional-de-convolutional networks for precise temporal action localization in untrimmed videos. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2017) 1417–1426

[108] Lea, C., Flynn, M.D., Vidal, R., Reiter, A., Hager, G.D.: Temporal convolutional networks for action segmentation and detection. arXiv preprint arXiv:1611.05267 (2016)

[109] Núñez-Marcos, A., Azkune, G., Arganda-Carreras, I.: Vision-based fall detection with convolutional neural networks. Wireless Communications and Mobile Computing **2017** (2017)

[110] Wang, S., Chen, L., Zhou, Z., Sun, X., Dong, J.: Human fall detection in surveillance video based on pcanet. Multimedia tools and applications **75** (2016) 11603–11613

[111] Babiker, H.K.B., Goebel, R.: An introduction to deep visual explanation. arXiv preprint arXiv:1711.09482 (2017)

[112] Ancona, M., Ceolini, E., Öztireli, C., Gross, M.: A unified view of gradient-based attribution methods for deep neural networks. arXiv preprint arXiv:1711.06104 (2017)

[113] Simonyan, K., Vedaldi, A., Zisserman, A.: Deep inside convolutional networks: Visualising image classification models and saliency maps. arXiv preprint arXiv:1312.6034 (2013)

[114] Sundararajan, M., Taly, A., Yan, Q.: Axiomatic attribution for deep networks. arXiv preprint arXiv:1703.01365 (2017)

[115] Shrikumar, A., Greenside, P., Kundaje, A.: Learning important features through propagating activation differences. arXiv preprint arXiv:1704.02685 (2017)

[116] Li, Y., Ye, Z., Rehg, J.M.: Delving into egocentric actions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2015) 287–295

[117] Ahmidi, N., Tao, L., Sefati, S., Gao, Y., Lea, C., Haro, B.B., Zappella, L., Khudanpur, S., Vidal, R., Hager, G.D.: A dataset and benchmarks for segmentation and recognition of gestures in robotic surgery. IEEE Transactions on Biomedical Engineering **64** (2017) 2025–2041

[118] Mathis, A., Mamidanna, P., Cury, K.M., Abe, T., Murthy, V.N., Mathis, M.W., Bethge, M.: Deeplabcut: markerless pose estimation of user-defined body parts with deep learning. Nature Neuroscience (2018) 1281 – 1289

[119] Ma, M., Fan, H., Kitani, K.M.: Going deeper into first-person activity recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2016) 1894–1903

[120] Wang, H., Kläser, A., Schmid, C., Liu, C.L.: Action recognition by dense trajectories. In: Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on, IEEE (2011) 3169–3176

[121] Laptev, I., Marszalek, M., Schmid, C., Rozenfeld, B.: Learning realistic human actions from movies. In: 2008 IEEE Conference on Computer Vision and Pattern Recognition. (2008) 1–8

[122] Wang, H., Kläser, A., Schmid, C., Liu, C.L.: Dense trajectories and motion boundary descriptors for action recognition. International journal of computer vision **103** (2013) 60–79

[123] Dalal, N., Triggs, B., Schmid, C.: Human detection using oriented histograms of flow and appearance. In: European conference on computer vision, Springer (2006) 428–441

[124] Laptev, I.: On space-time interest points. International journal of computer vision **64** (2005) 107–123

[125] Wang, H., Schmid, C.: Action recognition with improved trajectories. In: IEEE International Conference on Computer Vision (ICCV). (2013) 3551–3558

[126] Jégou, H., Douze, M., Schmid, C., Pérez, P.: Aggregating local descriptors into a compact image representation. In: CVPR 2010-23rd IEEE Conference on Computer Vision & Pattern Recognition, IEEE Computer Society (2010) 3304–3311

[127] Lowe, D.G.: Distinctive image features from scale-invariant keypoints. International journal of computer vision **60** (2004) 91–110

[128] Tran, D., Bourdev, L., Fergus, R., Torresani, L., Paluri, M.: Learning spatiotemporal features with 3d convolutional networks. In: Proceedings of the IEEE international conference on computer vision. (2015) 4489–4497

[129] Schölkopf, B., Smola, A.J.: Learning with kernels: Support vector machines, regularization, optimization, and beyond. MIT press (2002)

[130] von Luxburg, U.: A tutorial on spectral clustering. Statistics and Computing **17** (2007) 395–416

[131] Peng, R., Sun, H., Zanetti, L.: Partitioning well-clustered graphs: Spectral clustering works! SIAM Journal on Computing **46** (2017) 710–743

[132] Maaten, L.v.d., Hinton, G.: Visualizing data using t-sne. Journal of machine learning research **9** (2008) 2579–2605

[133] Messing, R., Pal, C., Kautz, H.: Activity recognition using the velocity histories of tracked keypoints. In: 2009 IEEE 12th International Conference on Computer Vision, IEEE (2009) 104–111

[134] Liberty, E., Sriharsha, R., Sviridenko, M.: An algorithm for online k-means clustering. In: Proceedings of the Eighteenth Workshop on Algorithm Engineering and Experiments (ALENEX). (2016) 81–89

[135] Kulis, B., Jordan, M.I.: Revisiting k-means: New algorithms via bayesian nonparametrics. arXiv preprint arXiv:1111.0352 (2011)

[136] Liu, L., Wang, L., Liu, X.: In defense of soft-assignment coding. In: 2011 International Conference on Computer Vision, IEEE (2011) 2486–2493

[137] Duchenne, O., Laptev, I., Sivic, J., Bach, F., Ponce, J.: Automatic annotation of human actions in video. In: 12th International Conference on Computer Vision, IEEE (2009) 1491–1498

[138] Gygli, M., Grabner, H., Riemenschneider, H., Van Gool, L.: Creating summaries from user videos. In: European conference on computer vision, Springer (2014) 505–520

[139] Gong, D., Medioni, G., Zhu, S., Zhao, X.: Kernelized temporal cut for online temporal segmentation and recognition. In: European conference on computer vision (ECCV). (2012) 229–243

[140] Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Transactions on pattern analysis and machine intelligence **22** (2000) 888–905

[141] Layher, G., Brosch, T., Neumann, H.: Real-time biologically inspired action recognition from key poses using a neuromorphic architecture. Frontiers in Neurorobotics **11** (2017) 13

[142] Huang, D., Yao, S., Wang, Y., De La Torre, F.: Sequential max-margin event detectors. In: European conference on computer vision (ECCV). (2014) 410–424

[143] Tenorth, M., Bandouch, J., Beetz, M.: The TUM Kitchen Data Set of Everyday Manipulation Activities for Motion Tracking and Action Recognition. In: IEEE International Workshop on Tracking Humans for the Evaluation of their Motion in Image Sequences (THEMIS), in conjunction with ICCV2009. (2009)

[144] Feichtenhofer, C., Pinz, A., Zisserman, A.: Convolutional two-stream network fusion for video action recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2016) 1933–1941

[145] Fleck, S., Straßer, W.: Smart camera based monitoring system and its application to assisted living. Proceedings of the IEEE **96** (2008) 1698–1714

[146] Nakashima, H., Aghajan, H., Augusto, J.C.: Handbook of ambient intelligence and smart environments. Springer Science & Business Media (2009)

[147] Demiröz, B.E., Ari, İ., Eroğlu, O., Salah, A.A., Akarun, L.: Feature-based tracking on a multi-omnidirectional camera dataset. In: 5th International Symposium on Communications Control and Signal Processing (ISCCSP), IEEE (2012) 1–5

[148] Mabrouk, A.B., Zagrouba, E.: Abnormal behavior recognition for intelligent video surveillance systems: A review. Expert Systems with Applications **91** (2018) 480–491

[149] Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: A survey. ACM computing surveys (CSUR) **41** (2009) 15

[150] He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: International Conference on Computer Vision (ICCV), IEEE (2017) 2980–2988

[151] Zhou, F., la Torre, F.D., Hodgins, J.K.: Aligned cluster analysis for temporal segmentation of human motion. In: 8th IEEE International Conference on Automatic Face and Gesture Recognition (FG). (2008) 1–7

[152] Mac, K.N.C., Joshi, D., Yeh, R.A., Xiong, J., Feris, R.R., Do, M.N.: Locally-consistent deformable convolution networks for fine-grained action detection. arXiv preprint arXiv:1811.08815 (2018)

[153] Li, Z., Gavrilyuk, K., Gavves, E., Jain, M., Snoek, C.G.: Videolstm convolves, attends and flows for action recognition. Computer Vision and Image Understanding **166** (2018) 41–50

[154] Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, Ieee (2009) 248–255

[155] Yosinski, J., Clune, J., Bengio, Y., Lipson, H.: How transferable are features in deep neural networks? In: Advances in neural information processing systems. (2014) 3320–3328

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

[157] Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. (2012) 1097–1105

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

[159] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2016) 2818–2826

[160] Szegedy, C., Ioffe, S., Vanhoucke, V., Alemi, A.A.: Inception-v4, inception-resnet and the impact of residual connections on learning. In: AAAI. Volume 4. (2017) 12

[161] Feichtenhofer, C., Pinz, A., Wildes, R.: Spatiotemporal residual networks for video action recognition. In: Advances in neural information processing systems. (2016) 3468–3476

[162] He, K., Girshick, R., Dollár, P.: Rethinking imagenet pre-training. arXiv preprint arXiv:1811.08883 (2018)

[163] Zhang, Y., Neumann, H.: An empirical study towards understanding how deep convolutional nets recognize falls. In: The European Conference on Computer Vision (ECCV) Workshops. (2018)

[164] Ma, X., Hovy, E.: End-to-end sequence labeling via bi-directional lstm-cnns-crf. arXiv preprint arXiv:1603.01354 (2016)

[165] Li, C., Zhang, Z., Lee, W.S., Lee, G.H.: Convolutional sequence to sequence model for human dynamics. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2018) 5226–5234

[166] Xu, B., Wang, N., Chen, T., Li, M.: Empirical evaluation of rectified activations in convolutional network. arXiv preprint arXiv:1505.00853 (2015)

[167] Dykes, P.C., Carroll, D.L., Hurley, A., Lipsitz, S., Benoit, A., Chang, F., Meltzer, S., Tsurikova, R., Zuyov, L., Middleton, B.: Fall prevention in acute care hospitals: a randomized trial. Jama **304** (2010) 1912–1918

[168] Gillain, S., Elbouz, L., Beaudart, C., Bruyère, O., Reginster, J., Petermans, J.: Falls in the elderly. Revue medicale de Liege **69** (2014) 258–264

[169] Zhang, T., Wang, J., Xu, L., Liu, P.: Fall detection by wearable sensor and one-class svm algorithm. In: Intelligent computing in signal processing and pattern recognition. Springer (2006) 858–863

[170] Wu, F., Zhao, H., Zhao, Y., Zhong, H.: Development of a wearable-sensor-based fall detection system. International journal of telemedicine and applications **2015** (2015) 2

[171] Cao, Z., Simon, T., Wei, S.E., Sheikh, Y.: Realtime multi-person 2d pose estimation using part affinity fields. In: CVPR. (2017)

[172] Insafutdinov, E., Andriluka, M., Pishchulin, L., Tang, S., Levinkov, E., Andres, B., Schiele, B.: Arttrack: Articulated multi-person tracking in the wild. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017) 1293–1301

[173] Kanazawa, A., Black, M.J., Jacobs, D.W., Malik, J.: End-to-end recovery of human shape and pose. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2018) 7122–7131

[174] Neverova, N., Wolf, C., Taylor, G.W., Nebout, F.: Multi-scale deep learning for gesture detection and localization. In: Workshop at the European conference on computer vision, Springer (2014) 474–490

[175] Graves, A., Fernández, S., Schmidhuber, J.: Bidirectional lstm networks for improved phoneme classification and recognition. In: International Conference on Artificial Neural Networks, Springer (2005) 799–804

[176] Maas, A.L., Hannun, A.Y., Ng, A.Y.: Rectifier nonlinearities improve neural network acoustic models. In: Proc. icml. Volume 30. (2013) 3

[177] Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)

[178] Charfi, I., Miteran, J., Dubois, J., Atri, M., Tourki, R.: Optimized spatio-temporal descriptors for real-time fall detection: comparison of support vector machine and adaboost-based classification. Journal of Electronic Imaging **22** (2013) 041106

[179] Chambolle, A.: An algorithm for total variation minimization and applications. Journal of Mathematical imaging and vision **20** (2004) 89–97

[180] Insafutdinov, E., Pishchulin, L., Andres, B., Andriluka, M., Schiele, B.: Deepercut: A deeper, stronger, and faster multi-person pose estimation model. In: European Conference on Computer Vision (ECCV). (2016) 34–50

[181] Tang, S., Andriluka, M., Andres, B., Schiele, B.: Multiple people tracking by lifted multicut and person re-identification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2017) 3539–3548

[182] Canny, J.: A computational approach to edge detection. IEEE Transactions on pattern analysis and machine intelligence (1986) 679–698

[183] Ben-Younes, H., Cadene, R., Cord, M., Thome, N.: Mutan: Multimodal tucker fusion for visual question answering. In: Proceedings of the IEEE international conference on computer vision. (2017) 2612–2620

[184] Bellman, R.E.: Adaptive control processes: a guided tour. Volume 2045. Princeton university press (2015)

[185] Golub, G.H., Van Loan, C.F.: Matrix computations. Volume 3. JHU Press (2012)

[186] Schölkopf, B., Smola, A.: Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press, Cambridge, MA, USA (2002)

[187] Muandet, K., Fukumizu, K., Sriperumbudur, B., Schölkopf, B.: Kernel mean embedding of distributions: A review and beyond. Foundations and Trends in Machine Learning **10** (2017) 1–141

[188] Kolda, T.G., Bader, B.W.: Tensor decompositions and applications. SIAM review **51** (2009) 455–500

[189] Kar, P., Karnick, H.: Random feature maps for dot product kernels. In: Artificial Intelligence and Statistics. (2012) 583–591

[190] Yu, F.X.X., Suresh, A.T., Choromanski, K.M., Holtmann-Rice, D.N., Kumar, S.: Orthogonal random features. In: Advances in Neural Information Processing Systems. (2016) 1975–1983

[191] Muirhead, R.J.: Aspects of multivariate statistical theory. Volume 197. John Wiley & Sons (2009)

[192] Parascandolo, G., Huttunen, H., Virtanen, T.: Taming the waves: sine as activation function in deep neural networks. (2016)

[193] Fathi, A., Ren, X., Rehg, J.M.: Learning to recognize objects in egocentric activities. In: Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference On, IEEE (2011) 3281–3288

[194] Gao, Y., Vedula, S.S., Reiley, C.E., Ahmidi, N., Varadarajan, B., Lin, H.C., Tao, L., Zappella, L., Béjar, B., Yuh, D.D., et al.: Jhu-isi gesture and skill assessment working set (jigsaws): A surgical activity dataset for human motion modeling. In: MICCAI Workshop: M2CAI. Volume 3. (2014) 3

[195] Nair, V., Hinton, G.E.: Rectified linear units improve restricted boltzmann machines. In: Proceedings of the 27th international conference on machine learning (ICML-10). (2010) 807–814

[196] Ramachandran, P., Zoph, B., Le, Q.V.: Searching for activation functions. CoRR **abs/1710.05941** (2017)

[197] Li, Y., Wang, N., Liu, J., Hou, X.: Factorized bilinear models for image recognition. In: Proceedings of the IEEE International Conference on Computer Vision. (2017) 2079–2087

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

[199] Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9** (1997) 1735–1780

[200] Delbrück, T., Linares-Barranco, B., Culurciello, E., Posch, C.: Activity-driven, event-based vision sensors. In: Proceedings of 2010 IEEE International Symposium on Circuits and Systems (ISCAS), IEEE (2010) 2426–2429

Appendix

1 Proof of Theorem 1

Proof. It follows from Eq. (4.28) and the property of an inner product of rank-one operators that

$$\begin{aligned} k(\boldsymbol{z}_1, \boldsymbol{z}_2) := \langle \boldsymbol{z}_1, \boldsymbol{z}_2 \rangle &= \left\langle \sum_{r=1}^{R} vec(\boldsymbol{E}^r \boldsymbol{x}_1 \otimes \boldsymbol{F}^r \boldsymbol{y}_1), \sum_{r=1}^{R} vec(\boldsymbol{E}^r \boldsymbol{x}_2 \otimes \boldsymbol{F}^r \boldsymbol{y}_2) \right\rangle_{\mathbb{R}^{MN}} \\ &= \sum_{r=1}^{R} \sum_{r'=1}^{R} \left\langle vec(\boldsymbol{E}^r \boldsymbol{x}_1 \otimes \boldsymbol{F}^r \boldsymbol{y}_1), vec(\boldsymbol{E}^{r'} \boldsymbol{x}_2 \otimes \boldsymbol{F}^{r'} \boldsymbol{y}_2) \right\rangle_{\mathbb{R}^{MN}} \\ &= \sum_{r=1}^{R} \sum_{r'=1}^{R} \left\langle \boldsymbol{E}^r \boldsymbol{x}_1, \boldsymbol{E}^{r'} \boldsymbol{x}_2 \right\rangle \left\langle \boldsymbol{F}^r \boldsymbol{y}_1, \boldsymbol{F}^{r'} \boldsymbol{y}_2 \right\rangle \\ &= \sum_{r=1}^{R} \sum_{r'=1}^{R} \left(\sum_{i=1}^{M} \langle \boldsymbol{e}_i^r, \boldsymbol{x}_1 \rangle \langle \boldsymbol{e}_i^{r'}, \boldsymbol{x}_2 \rangle \right) \left(\sum_{j=1}^{N} \langle \boldsymbol{f}_j^r, \boldsymbol{y}_1 \rangle \langle \boldsymbol{f}_j^{r'}, \boldsymbol{y}_2 \rangle \right). \end{aligned} \quad (1)$$

Then, it follows that

$$\begin{aligned} \mathbb{E}[k(\boldsymbol{z}_1, \boldsymbol{z}_2)] &= \sum_{r=1}^{R} \sum_{r'=1}^{R} \left(\sum_{i=1}^{M} \mathbb{E}[\langle \boldsymbol{e}_i^r, \boldsymbol{x}_1 \rangle \langle \boldsymbol{e}_i^{r'}, \boldsymbol{x}_2 \rangle] \right) \left(\sum_{j=1}^{N} \mathbb{E}[\langle \boldsymbol{f}_j^r, \boldsymbol{y}_1 \rangle \langle \boldsymbol{f}_j^{r'}, \boldsymbol{y}_2 \rangle] \right) \\ &= \sum_{r=1}^{R} \left(\sum_{i=1}^{M} \mathbb{E}[\langle \boldsymbol{e}_i^r, \boldsymbol{x}_1 \rangle \langle \boldsymbol{e}_i^r, \boldsymbol{x}_2 \rangle] \right) \left(\sum_{j=1}^{N} \mathbb{E}[\langle \boldsymbol{f}_j^r, \boldsymbol{y}_1 \rangle \langle \boldsymbol{f}_j^r, \boldsymbol{y}_2 \rangle] \right) \\ &\quad + \sum_{r=1}^{R} \sum_{r'=r+1}^{R} \left(\sum_{i=1}^{M} \mathbb{E}[\langle \boldsymbol{e}_i^r, \boldsymbol{x}_1 \rangle \langle \boldsymbol{e}_i^{r'}, \boldsymbol{x}_2 \rangle] \right) \left(\sum_{j=1}^{N} \mathbb{E}[\langle \boldsymbol{f}_j^r, \boldsymbol{y}_1 \rangle \langle \boldsymbol{f}_j^{r'}, \boldsymbol{y}_2 \rangle] \right) \\ &= RMN \langle \boldsymbol{x}_1, \boldsymbol{x}_2 \rangle \langle \boldsymbol{y}_1, \boldsymbol{y}_2 \rangle. \end{aligned}$$

The last equation follows from [189, Lemma 2] and the fact that $\boldsymbol{e}_i^r$ and $\boldsymbol{f}_j^r$ are zero-mean random variables for all $i = 1, \ldots, M$, $j = 1, \ldots, N$ and $r = 1, \ldots, R$. □

2 Proof of Corollary 1

Proof. Let $W_{ij} := \langle \boldsymbol{e}_i, \boldsymbol{x}_1 \rangle \langle \boldsymbol{e}_i, \boldsymbol{x}_2 \rangle \langle \boldsymbol{f}_j, \boldsymbol{y}_1 \rangle \langle \boldsymbol{f}_j, \boldsymbol{y}_2 \rangle$ for $i = 1, \ldots, M$ and $j = 1, \ldots, N$. For each W_{ij}, it follows from [189, Lemma 4] that

$$-p^2 f(p\tilde{R}^2)^2 \leq W_{ij} \leq p^2 f(p\tilde{R}^2)^2,$$

where we assume without loss of generality that $p \geq 1$ and $\tilde{R} \geq 1$. In our case, $f(x) = x$. Then, we have $-p^4\tilde{R}^4 \leq W_{ij} \leq p^4\tilde{R}^4$. Let $S_{MN} := \frac{1}{R}\sum_{i=1}^{M}\sum_{j=1}^{N} W_{ij}$. Hence, we have $\mathbb{E}[S_{MN}] = MN\langle \boldsymbol{x}_1, \boldsymbol{x}_2 \rangle \langle \boldsymbol{y}_1, \boldsymbol{y}_2 \rangle$. Then, it follows from Hoeffding's inequality that, for all $\epsilon > 0$,

$$\mathbb{P}(|S_{MN} - \mathbb{E}[S_{MN}]| \geq \epsilon) \leq 2\exp\left(-\frac{\epsilon^2 MN}{2p^8\tilde{R}^8}\right). \tag{2}$$

This concludes the proof. □

3 Proof of Theorem 2

Proof. Let $R = 1$. Then, $k(\boldsymbol{z}_1, \boldsymbol{z}_2) := \langle \boldsymbol{z}_1, \boldsymbol{z}_2 \rangle = \langle \varphi(\boldsymbol{E}\boldsymbol{x}_1), \varphi(\boldsymbol{E}\boldsymbol{x}_2) \rangle \langle \varphi(\boldsymbol{F}\boldsymbol{x}_1), \varphi(\boldsymbol{F}\boldsymbol{x}_2) \rangle$ where

$$\begin{aligned}
\varphi(\boldsymbol{E}\boldsymbol{x}) &:= \frac{1}{\sqrt{M}}[\sin(\boldsymbol{e}_1^\top \boldsymbol{x}), \ldots, \sin(\boldsymbol{e}_M^\top \boldsymbol{x}), \cos(\boldsymbol{e}_1^\top \boldsymbol{x}), \ldots, \cos(\boldsymbol{e}_M^\top \boldsymbol{x})], \\
\varphi(\boldsymbol{F}\boldsymbol{y}) &:= \frac{1}{\sqrt{N}}[\sin(\boldsymbol{f}_1^\top \boldsymbol{y}), \ldots, \sin(\boldsymbol{f}_M^\top \boldsymbol{y}), \cos(\boldsymbol{f}_1^\top \boldsymbol{y}), \ldots, \cos(\boldsymbol{f}_M^\top \boldsymbol{y})].
\end{aligned}$$

With $\boldsymbol{E} = \frac{1}{\sigma}\boldsymbol{I}_{M \times D_X}\boldsymbol{R}\boldsymbol{P}$ and $\boldsymbol{F} = \frac{1}{\rho}\boldsymbol{I}_{N \times D_Y}\boldsymbol{S}\boldsymbol{Q}$, we have

$$\begin{aligned}
\mathbb{E}[k(\boldsymbol{z}_1, \boldsymbol{z}_2)] &= \mathbb{E}[\langle \varphi(\boldsymbol{E}\boldsymbol{x}_1), \varphi(\boldsymbol{E}\boldsymbol{x}_2) \rangle]\mathbb{E}[\langle \varphi(\boldsymbol{F}\boldsymbol{y}_1), \varphi(\boldsymbol{F}\boldsymbol{y}_2) \rangle] \\
&= \exp\left(-\frac{\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2^2}{2\sigma^2}\right)\exp\left(-\frac{\|\boldsymbol{y}_1 - \boldsymbol{y}_2\|_2^2}{2\rho^2}\right),
\end{aligned}$$

where the last equality follows from [190, Theorem 1]. For $R > 1$, we have

$$k(\boldsymbol{z}_1, \boldsymbol{z}_2) := \langle \boldsymbol{z}_1, \boldsymbol{z}_2 \rangle = \sum_{r=1}^{R}\sum_{r'=1}^{R} \langle \varphi(\boldsymbol{E}^r \boldsymbol{x}_1), \varphi(\boldsymbol{E}^{r'} \boldsymbol{x}_2) \rangle \langle \varphi(\boldsymbol{F}^r \boldsymbol{x}_1), \varphi(\boldsymbol{F}^{r'} \boldsymbol{x}_2) \rangle.$$

Hence, with $\boldsymbol{E}^r = \frac{1}{\sigma^r}\boldsymbol{I}_{M\times D_X}\boldsymbol{R}^r\boldsymbol{P}^r$ and $\boldsymbol{F}^r = \frac{1}{\rho^r}\boldsymbol{I}_{N\times D_Y}\boldsymbol{S}^r\boldsymbol{Q}^r$,

$$\begin{aligned}
\mathbb{E}[k(\boldsymbol{z}_1,\boldsymbol{z}_2)] &= \sum_{r=1}^{R}\sum_{r'=1}^{R}\mathbb{E}[\langle\varphi(\boldsymbol{E}^r\boldsymbol{x}_1),\varphi(\boldsymbol{E}^{r'}\boldsymbol{x}_2)\rangle]\mathbb{E}[\langle\varphi(\boldsymbol{F}^r\boldsymbol{y}_1),\varphi(\boldsymbol{F}^{r'}\boldsymbol{y}_2)\rangle] \\
&= \sum_{r=1}^{R}\mathbb{E}[\langle\varphi(\boldsymbol{E}^r\boldsymbol{x}_1),\varphi(\boldsymbol{E}^r\boldsymbol{x}_2)\rangle]\mathbb{E}[\langle\varphi(\boldsymbol{F}^r\boldsymbol{y}_1),\varphi(\boldsymbol{F}^r\boldsymbol{y}_2)\rangle] \\
&+ \sum_{r=1}^{R}\sum_{r'=r+1}^{R}\mathbb{E}[\langle\varphi(\boldsymbol{E}^r\boldsymbol{x}_1),\varphi(\boldsymbol{E}^{r'}\boldsymbol{x}_2)\rangle]\mathbb{E}[\langle\varphi(\boldsymbol{F}^r\boldsymbol{y}_1),\varphi(\boldsymbol{F}^{r'}\boldsymbol{y}_2)\rangle] \\
&= \sum_{r=1}^{R}\exp\left(-\frac{\|\boldsymbol{x}_1-\boldsymbol{x}_2\|_2^2}{2\sigma_r^2}\right)\exp\left(-\frac{\|\boldsymbol{y}_1-\boldsymbol{y}_2\|_2^2}{2\rho_r^2}\right) \\
&+ \sum_{r=1}^{R}\sum_{r'=r+1}^{R}\mathbb{E}[\langle\varphi(\boldsymbol{E}^r\boldsymbol{x}_1),\varphi(\boldsymbol{E}^{r'}\boldsymbol{x}_2)\rangle]\mathbb{E}[\langle\varphi(\boldsymbol{F}^r\boldsymbol{y}_1),\varphi(\boldsymbol{F}^{r'}\boldsymbol{y}_2)\rangle] \\
&= \sum_{r=1}^{R}\exp\left(-\frac{\|\boldsymbol{x}_1-\boldsymbol{x}_2\|_2^2}{2\sigma_r^2}\right)\exp\left(-\frac{\|\boldsymbol{y}_1-\boldsymbol{y}_2\|_2^2}{2\rho_r^2}\right) + b
\end{aligned}$$

where $b := \sum_{r=1}^{R}\sum_{r'=r+1}^{R}\mathbb{E}[\langle\varphi(\boldsymbol{E}^r\boldsymbol{x}_1),\varphi(\boldsymbol{E}^{r'}\boldsymbol{x}_2)\rangle]\mathbb{E}[\langle\varphi(\boldsymbol{F}^r\boldsymbol{y}_1),\varphi(\boldsymbol{F}^{r'}\boldsymbol{y}_2)\rangle]$. □

4 Approximation error bound for Gaussian random projection

We characterize the variance of $k(\boldsymbol{z}_1,\boldsymbol{z}_2)$ used in Theorem 2. To simplify the presentation, we focus on the case that $R = 1$.

Corollary 2. *Let $\boldsymbol{z}_1$, $\boldsymbol{z}_2$, and $k(\boldsymbol{z}_1,\boldsymbol{z}_2)$ be defined as in Theorem 2, as well as $R = 1$, $a = \|\boldsymbol{x}_1-\boldsymbol{x}_2\|_2^2/\sigma$ and $b = \|\boldsymbol{y}_1-\boldsymbol{y}_2\|_2^2/\rho$. Then, there exist functions f and g such that*

$$\mathrm{Var}(k(\boldsymbol{z}_1,\boldsymbol{z}_2)) \leq A\cdot B + A\cdot C + B\cdot D, \tag{3}$$

where

$$A = \frac{1}{2M}\left[\left(\left(1-e^{-a^2}\right)^2 - \frac{M-1}{D_X}e^{-a^2}a^4\right) + \frac{f(a)}{D_X^2}\right], \qquad C = \left[\exp\left(-\frac{b^2}{2}\right)\right]^2$$
$$B = \frac{1}{2N}\left[\left(\left(1-e^{-b^2}\right)^2 - \frac{N-1}{D_Y}e^{-b^2}b^4\right) + \frac{g(b)}{D_Y^2}\right], \qquad D = \left[\exp\left(-\frac{a^2}{2}\right)\right]^2.$$

Proof. Let R=1. Then, we have

$$k(\boldsymbol{z}_1, \boldsymbol{z}_2) = \underbrace{\langle\varphi(\boldsymbol{E}\boldsymbol{x}_1), \varphi(\boldsymbol{E}\boldsymbol{x}_2)\rangle}_{U}\underbrace{\langle\varphi(\boldsymbol{F}\boldsymbol{x}_1), \varphi(\boldsymbol{F}\boldsymbol{x}_2)\rangle}_{V}.$$

Since U and V are independent, we have

$$\mathrm{Var}(k(\boldsymbol{z}_1, \boldsymbol{z}_2)) = \mathrm{Var}(U)\mathrm{Var}(V) + \mathrm{Var}(U)(\mathbb{E}[V])^2 + \mathrm{Var}(V)(\mathbb{E}[U])^2, \tag{4}$$

where

$$(\mathbb{E}[U])^2 = (\mathbb{E}\left[\langle\varphi(\boldsymbol{E}\boldsymbol{x}_1), \varphi(\boldsymbol{E}\boldsymbol{x}_2)\rangle\right])^2 \qquad (\mathbb{E}[V])^2 = (\mathbb{E}\left[\langle\varphi(\boldsymbol{F}\boldsymbol{y}_1), \varphi(\boldsymbol{F}\boldsymbol{y}_2)\rangle\right])^2$$
$$\mathrm{Var}(U) = \mathrm{Var}\left(\langle\varphi(\boldsymbol{E}\boldsymbol{x}_1), \varphi(\boldsymbol{E}\boldsymbol{x}_2)\rangle\right) \qquad \mathrm{Var}(V) = \mathrm{Var}\left(\langle\varphi(\boldsymbol{F}\boldsymbol{y}_1), \varphi(\boldsymbol{F}\boldsymbol{y}_2)\rangle\right)$$

It follows from [190, Theorem 1] that

$$(\mathbb{E}[U])^2 = \left(\exp\left(-\frac{\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2^2}{2\sigma^2}\right)\right)^2, \quad (\mathbb{E}[V])^2 = \left(\exp\left(-\frac{\|\boldsymbol{y}_1 - \boldsymbol{y}_2\|_2^2}{2\rho^2}\right)\right)^2.$$

Let $a = \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2^2/\sigma$ and $b = \|\boldsymbol{y}_1 - \boldsymbol{y}_2\|_2^2/\rho$. Then, by [190, Theorem 1], there exists a function f and g such that

$$\mathrm{Var}(U) \leq \frac{1}{2M}\left[\left(\left(1-e^{-a^2}\right)^2 - \frac{M-1}{D_X}e^{-a^2}a^4\right) + \frac{f(a)}{D_X^2}\right]$$
$$\mathrm{Var}(V) \leq \frac{1}{2N}\left[\left(\left(1-e^{-b^2}\right)^2 - \frac{N-1}{D_Y}e^{-b^2}b^4\right) + \frac{g(b)}{D_Y^2}\right].$$

Substituting everything back into (4) yields the result. □

List of Figures

List of Tables

Name: Yan Zhang
aus Qingzhou, Shandong, China Matriculation number: 1006669

Honesty disclaimer

I hereby affirm that I wrote this thesis independently and that I did not use any other sources or tools than the ones specified.

Ulm, ...

Yan Zhang

aus Qingzhou, Shandong, China