Discriminative Video Representation Learning

Australian National University

Jue Wang

Research School of Engineering
College of Engineering and Computer Science
Australian National University

This dissertation is submitted for the degree of
Doctor of Philosophy

 July 2022

I would like to dedicate this thesis to my loving parents and wife.

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements.

Jue Wang
July 2022

Acknowledgements

There are many people whom I would like to thank for their contribution, both directly and indirectly, to this thesis.

Foremost, I would like to express my sincere gratitude to my panel members, Prof. Fatih Porikli, Prof. Richard Hartley and Dr. Anoop Cherian. I am indebted to Prof. Fatih Porikli for having me as his Ph.D student, leading me into the world of computer vision. His encouragement, insightful comments and patience inspire me in many ways. I must express my special thanks to my primary supervisor, Dr. Anoop Cherian for his continuous support of my Ph.D study and research. His patience, motivation, enthusiasm, valuable comments and immense knowledge helped me in all the time of research and writing of this thesis. I greatly appreciate the internship opportunity he provided in the Mitsubishi Electric Research Laboratories, which helped me a lot in building my career path. Dr. Anoop Cherian is definitely the best supervisor and mentor I could have imagined. It is also my great honour to have Prof. Richard Hartley in my research panel. Discussion with him could always enlighten me with fresh and novel ideas.

In the last few years, I am fortunate enough to work with many brilliant researchers all around the world. I would like to express my gratitude to Prof. Yuchao Dai and Dr. Lars Petersson from the Australian National University for guiding me to do research projects when I was a undergraduate student. These unforgettable experiences established my research interest in the early stage. I would like to thank Dr. Chiori Hori, Dr. Tim K. Marks, Prof. Chen Feng and Prof. Devi Parikh for their help during my internship in the Mitsubishi Electric Research Laboratories. Their timely suggestions, dynamism and enthusiasm have enabled me to complete exciting research topics. I also owe my thanks to Prof. Lorenzo Torresani for providing me a chance to visit the Facebook AI Research Lab. It is always interesting and inspiring to meet with him to exchange ideas and opinions.

I would like to thank the Australian National University and National ICT Australia Ltd for providing me Ph.D scholarship and funding me to attend international conferences. I also thank the Australian Centre for Robotic Vision for organising academic seminars for sharing ideas and discussing novel methodologies. It is also a great pleasure to thank my

dear friends: Dr. Hongtao Yang, Dr. Tong Zhang, Mr. Yuexi Zhang, Mr. Huizhong Deng and Ms. Sarah Wu, for their support and encouragement in my life.

At last but not least, I would like to thank my parents, Lianping Wang and Yaping Wang and my grandfather, Meizhi Wang, for their unconditioned love and support in my Ph.D study. I especially want to thank my mother, Yaping Wang, who has been fighting against cancer for 12 years. Life is not fair to her, but she never gives up. It was her bravery and optimism in the face of illness inspired me to conquer obstacles in my life. No challenges in my Ph.D research could compare to what she has gone through in the last 12 years.

It is my privilege to thank my wife, Yajie Guan, for her love and consistent encouragement throughout my research period. She is always my first reader and audience, inspiring me with priceless feedback and suggestions. Her support and understanding was in the end what made this thesis possible.

This list of acknowledgements can only capture a small fraction of the people who supported my work. I would also want to thank everyone who was not named here but helped make this thesis a success in any manner.

Abstract

Representation learning is a fundamental research problem in the area of machine learning, refining the raw data to discover representations needed for various applications. However, real-world data, particularly video data, is neither mathematically nor computationally convenient to process due to its semantic redundancy and complexity. Video data, as opposed to images, includes temporal correlation and motion dynamics, but the ground truth label is normally limited to category labels, which makes the video representation learning a challenging problem. To this end, this thesis addresses the problem of video representation learning, specifically discriminative video representation learning, which focuses on capturing useful data distributions and reliable feature representations improving the performance of varied downstream tasks. We argue that neither all frames in one video nor all dimensions in one feature vector are useful and should be equally treated for video representation learning. Based on this argument, several novel algorithms are investigated in this thesis under multiple application scenarios, such as action recognition, action detection and one-class video anomaly detection. These proposed video representation learning methods produce discriminative video features in both deep and non-deep learning setups. Specifically, they are presented in the form of: 1) an early fusion layer that adopts a temporal ranking SVM formulation, agglomerating several optical flow images from consecutive frames into a novel compact representation, named as dynamic optical flow images; 2) an intermediate feature aggregation layer that applies weakly-supervised contrastive learning techniques, learning discriminative video representations via contrasting positive and negative samples from a sequence; 3) a new formulation for one-class feature learning that learns a set of discriminative subspaces with orthonormal hyperplanes to flexibly bound the one-class data distribution using Riemannian optimisation methods. We provide extensive experiments to gain intuitions into why the learned representations are discriminative and useful. All the proposed methods in this thesis are evaluated on standard publicly available benchmarks, demonstrating state-of-the-art performance.

Keywords: Video representation learning, discriminative pooling, action recognition, temporal modelling, contrastive learning.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

1.1 Overview

Vision is one of our most important senses that enable us to see this colourful world. Through our vision, we perceive surroundings, learn the knowledge and make decisions. As a result, vision data is normally treated as one of the most important input sources in the area of artificial intelligence, specifically, machine learning. However, teaching computers to see as we see and understand this world as we understand is a challenging topic. To this end, the discipline of computer vision is proposed, which is a sub-area of computer science that focuses on creating digital systems that can process, analyse, and make sense of visual data (images or videos) in the same way that humans do. Theoretically, the concept of computer vision involves knowledge of signal processing, artificial intelligence, neurobiology, solid-state physics and robotics. In practice, it is based on teaching computers to process the visual data at pixel level and understand it. An illustration of human vision system and computer vision system is shown in Figure 1.1. However, the raw pixel-level data is too large for computers to process. For example, each camera-captured image is formed by thousands or millions of pixels (which depends on the resolution), and each pixel is consisted of three digital numbers representing three primary colors: Red, Green and Blue. A 256 x 256 picture includes 200 thousands of numbers in the digital world. When it comes to video data, a video clip that lasts one second normally include 10 million pixels and if this video clip is 4K resolution, this number will be increased to 1 billion. By contrast, humans will not look into those 1 billion numbers every second when seeing 4K movies. This is because we are able to extract meaningful features and high-level semantic information instantly when we see the visual data. As a result, how to represent the raw visual data into a compact and computer-friendly way is key research topic in machine learning, commonly known as *representation learning* [9].

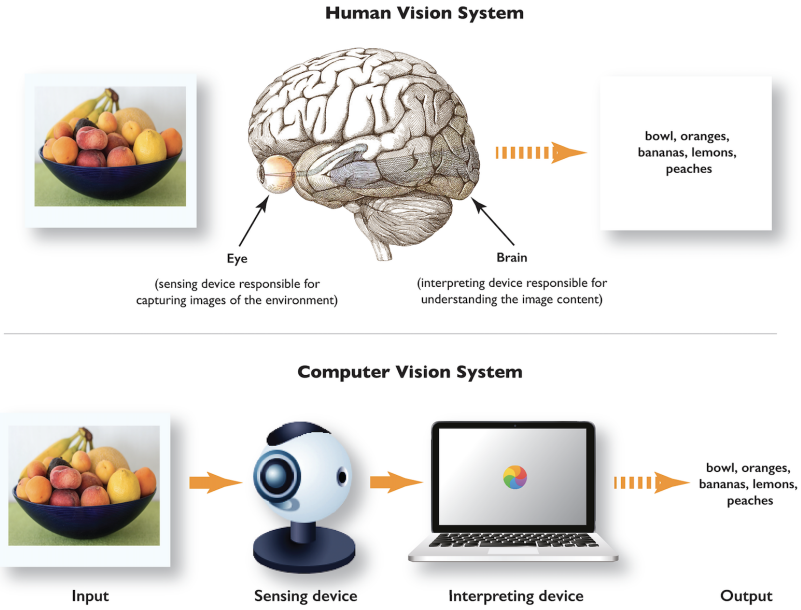


Fig. 1.1 Illustration of the human and computer vision systems [56].

In the area of computer vision, representation learning normally couples with various computer vision applications. Depending on the input data type, representation learning can be classified as image representation learning and video representation learning. In practice, representations are learned to solve one specific task, such as recognition, detection, generation, and anticipation for images or videos. As a result, it not only should compress the raw visual data, but also need to capture the meaningful features and the distribution from the raw data. For example, to distinguish bananas and apples, the shape feature would play a key role. However, when comparing oranges and apples, the color distribution is of more importance. Those features that help to produce correct answers are called discriminative information. The representation learning method that places extra effort on learning discriminative information is named as discriminative representation learning. In the topic of representation learning, it is ideal to learn discriminative representation, which leads to outstanding performance in given tasks.

In this thesis, the research focus on the video data and its applications such as action recognition and action detection. Specifically, given a set of videos, this thesis aims at building machine learning algorithms to learn discriminative video representations, which are then used to accomplish video-based tasks, such as predicting action categories (recognition) and locations (detection). In addition, this thesis focuses on supervised and weakly-supervised machine learning techniques, in which the ground truth label will be available or partly

available during the training. To simulate real world scenarios, video benchmarks with different configurations are used to evaluate the performance of algorithms in this thesis.

1.2 Video Representation learning

As mentioned above, the representation learning can be classified into image representation learning and video representation learning depending on the input type. Unlike describing image data, video representations not only should involve object information in each frame but also need to aggregate motion dynamics. Compared to the image representation learning, the video representation learning is far more challenging due to 1) the redundancy nature in the video sequence, and 2) complex motion dynamics without adequate supervision signal. The details of these two challenges will be explained as following. 1) Redundancy nature: As mentioned earlier, each video clip is consisted of hundreds or thousands of images, which dramatically increases the computational cost. In addition, the content of video data is highly repeated across frames and it is hard to summarize the discriminative information in it. 2) Inadequate supervision signal: The ground truth information for video data involves action and object categories, temporal locations, and skeleton coordinates. However, temporal correlations, motion flows and the way of how to present motion dynamics vary in each video and such information is normally fragile due to the arbitrary length of motions, camera movement, unexpected scene changes, occlusion and so on. Thus, these uncertainties make extracting spatiotemporal semantic information considerably more difficult than learning image representations. To figure out how to tackle these challenges, it is important to review previous works and identify what is the discriminative information in the video representation learning.

Researchers initially extract local hand-crafted features from video clips to represent motion dynamics in the video data. Using local features to generate representations for videos is popular because they make the recognition robust to noise, background motion, or illumination changes. One noteworthy such representation is described in [200], where dense trajectories, that capture the dynamics of actions, is used to define regions of interest in the video. Local features (such as HOG, HOF, MBH, etc.), that directly relate to the action can then be extracted from these regions to train classifiers. Specifically, the HOF is the histogram of optical flow, which capture the pixel movement between two adjacent frames; the HOG represent the histogram of gradient, capturing the color changes inside each frame, and the MBH(motion boundary histograms) is to compute derivatives separately for the horizontal and vertical components of the optical flow. Figure 1.2 demonstrates a visualization of these low-level hand crafted feature descriptors. However, the recent trend is

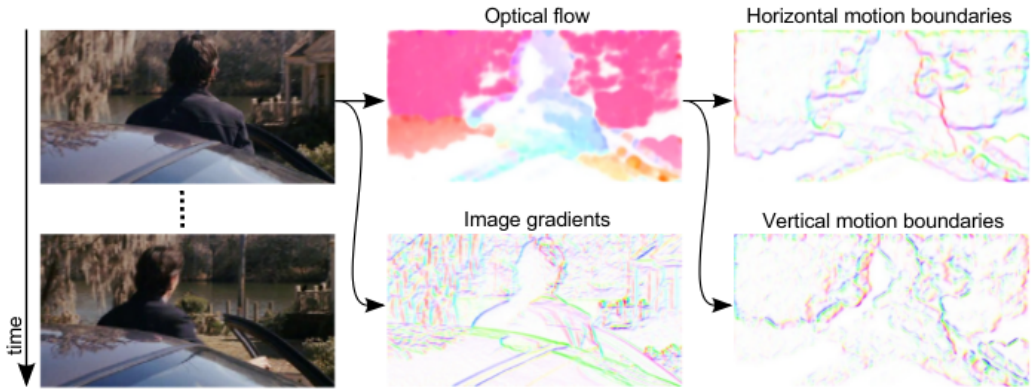


Fig. 1.2 Illustration of the hand crafted features such as HOG, HOF, and MBH descriptors [200].

towards automatically learning useful features in a data-driven way via convolutional neural networks [13, 62, 209], which is also the focus of this thesis.

With the huge success of deep learning schemes in the image recognition [106], deep learning methods have been extensively used for computer vision tasks. Also, the deep feature gradually becomes the main stream of current video representation learning algorithms. Based on the model in [106], there have been extensions for the problem of video representation learning. Most of which follow the supervised training fashion and offer innovations on capturing temporal evolution, as temporal dynamics and correlation are believed to be the trigger for discriminative representation learning. In additional to the RBG data, other modalities are also generated from the video: they are: 1) RGB difference, which subtract two adjacent frames, 2) optical flow, that calculate the pixel movement between two frames. Based on these, one of successful works is two-stream CNN model [170], which proposes a temporal stream to learn temporal dynamics by taking stacked optical flow images. As an initial attempt, two stream model and its variants [62, 211] successfully extend the regular image based CNN into the temporal space by using the optical flow. The video representation from two-modalities also become popular in various works with different CNN architectures [204, 202, 209]. Another direction is to implement recurrent neural networks [5, 53] and long-short term memory (LSTM) networks [115, 226], that processes the time series data step by step to capture the temporal dependency. These neural works were initially designed to process the sequential data such as audio and language model. After combining it with the CNN, it achieves promising performance in different applications. In particular, the LSTM successfully addresses the challenge of capturing long-term memory, which avoids gradient disappearing or explosion by using forget gate. Different from the input and network architecture innovation, another alternative is 3D convolutional neural network [188], which

is proposed as a direct extension of image-based architecture. Instead of taking 2D image patch, 3D CNN takes a 3D volume from the video clip to learn the spatiotemporal feature. 3D CNN is a straightforward way to capture temporal relationship by introducing more parameters. To make it more efficient, several variant versions are proposed, such as inflating the 2D convolutional kernel [24] and decoupling the spatial and temporal convolution [189].

1.3 Motivation

In the last section, previous video representation learning methods are summarized, which facilitate the discussion below.

Discriminative information in the video data involves both motion dynamics and object information. It is critical to understand how to properly encode the temporal evolution of videos in order to have discriminative video representation. To address this, previous methods propose innovation in three different perspectives: 1) from the input side, which includes calculating low-level hand-crafted feature and using multi-modalities in the deep neural network. 2) upgrading the CNN architecture, which includes using the recurrent neural network, LSTM and GRU for video representation learning. 3) changing the basic computing unit in CNNs while keeping CNNs architecture as the same, such as 3D CNNs and its variants.

However, every coin has two sides. Each of these methods cannot avoid introducing one or more disadvantages. For example, two-stream model and low-level hand-crafted features capture the temporal dynamics from additional modalities, but this significantly increases the computational cost which becomes impractical in large-scaled video datasets. Similarly, both recurrent network and 3D CNNs introduce more parameters which require a large number of training data to achieve top performance. Unlike natural language, video data includes more interference such as speed and occlusion. Even for videos with the same ground truth label, most of those information is not consistent, which make the learning process more complex. This is another reason RNNs and 3D CNNs require more training data.

Moreover, video sequences have arbitrarily length: it is impossible to feed entire sequences into the memory for learning video representations. Previous techniques reduce the duration of each video into numerous video clips of much shorter lengths. Then, they train the model with clip-level supervision and aggregate the prediction of clips to form the video-level prediction during the inference. Because the video clip would only cover a portion of motion dynamics, this may degrade the quality of the video representation. In addition, global average pooling will impose equal importance on all clips from each video and all dimensions of each feature vector. This might not be favourable as not all

frames/dimensions may characterize the underlying discriminative information. In summary, we argue that not all frames in one video and not all dimensions in one feature vector are useful for discriminative representation learning. It is worth exploring alternative pooling strategies to substitute current aggregation schemes in the video representation learning.

This thesis aims to address several limitations in the current video representation learning methods, and propose the following contributions for handling wider scope of real-world scenarios and improve the video feature quality, producing discriminative video representation.

1.4 Contributions

In this thesis, various pooling schemes are proposed to aggregate frame-level features into more compact representations while capturing the discriminative information, that benefit the target video-related tasks. The contributions of this thesis are summarized as follows:

- We provide an efficient and powerful video representation, *dynamic flow images*, that is generated based on the ranking SVM formulation. This representation aggregates local action dynamics (as captured by optical flow) over subseqeunces while preserving the temporal evolution of these dynamics. Associated with the popular two-stream network [170], a three-stream architecture for action recognition is proposed, demonstrating the effectiveness of this scheme on two challenging benchmark datasets.

  This work has been published at WACV 2017 [205]

- We introduce the concept of multiple instance learning (MIL) into a binary SVM classification problem for learning video descriptors, named as SVM pooling. It summarizes discriminative equivalent while explicitly encoding the action dynamics from video sequences. To increase the efficiency, we explore variants of proposed optimization method and present progressively cheaper inference schemes, including a joint pooling and classification objective, as well as an end-to-end learnable CNN architecture. At last, the usefulness of proposed video descriptors is demonstrated by applying it on eight popular vision benchmarks spanning diverse input data modalities and CNN architectures.

  Part of this work has been published at CVPR 2018 [206] and its extension has been published at TPAMI [202]

- To improve SVM pooling with robust negative instances, we introduce adversarial perturbations into the video recognition setting for weakly-supervised contrastively

learning robust video representations from pre-trained network. A binary classification problem is formulated to learn temporally-ordered discriminative subspaces that separate the data features from their perturbed counterparts. Each subspace is consisted of multiple orthogonal hyperplanes to avoid redundancy. In addition, efficient Riemannian optimization schemes are also provided for solving the objective on the Stiefel manifold.

This work has been published at ECCV 2018 [201]

- To extend the concept of learning video representations with hyperplanes, we propose a basic one-class discriminative subspace (BODS) classifier, which uses a pair of hyperplanes to characterize one-class data distribution. Based on this, BODS is then generalized to use multiple hyperplanes, termed generalized one-class discriminative subspaces (GODS) and derive a kernelized variant, termed KODS. Several formulations of GODS are also presented under different assumptions on the classifiers, in which we explore Riemannian conjugate gradient algorithms for optimizing proposed objectives. Specifically, the BODS and GODS formulations use a Stiefel manifold, while KODS is modeled on the generalized Stiefel manifold. At last, a novel task of out-of-pose detection is presented, associated with a new video dataset, termed Dash-Cam-Pose.

  Part of this work has been published at ICCV 2019 [203] and its extension has been published at TPAMI [36]

1.5 Thesis outline

Following this introduction chapter, the remaining chapters of this thesis are organized as below:

In Chapter 2, we provide literature review, introducing the basic concept of convolutional neural networks and its application in the area of computer vision. After that, the evolution of video feature learning in the recent decades is demonstrated, which includes their background, development and frameworks. At last, the popular pooling schemes in the video data are reviewed, pointing out the issues that are still unsolved.

In Chapter 3, a novel ordered representation is introduced, which consists of consecutive optical flow frames. It is argued that this representation captures the action dynamics more effectively than RGB frames. Intuitions on why such a representation is better for action recognition is also presented. Claims are validated on standard benchmark datasets, demonstrating that using summaries of flow images lead to significant improvements over RGB frames.

In Chapter 4, discriminative pooling is proposed, which is based on the notion that among the deep features generated on all short clips, there is at least one that characterizes the action. To identify these useful features, a negative bag consisting of features that are known to be irrelevant is picked, for example, they are sampled either from datasets that are unrelated to actions of interest or are CNN features produced via random noise as input. With the features from the video as a positive bag and the irrelevant features as the negative bag, an objective is proposed to learn a (nonlinear) hyperplane that separates the unknown useful features from the rest in a multiple instance learning formulation within a support vector machine setup. The parameters of this separating hyperplane are used as a descriptor for the full video segment. Since these parameters are directly related to the support vectors in a max-margin framework, they can be treated as a weighted average pooling of the features from the bags, with zero weights given to non-support vectors. This pooling scheme is end-to-end trainable within a deep learning framework.

In Chapter 5, universal perturbations is proposed to use within a novel contrastive learning setup to build negative samples, which are then used to produce improved video representations. To this end, given a pre-trained deep model for per-frame video recognition, adversarial noise is first generated to adapt to this model. Positive and negative bags are produced using the original data features from the full video sequence and their perturbed counterparts, respectively. Unlike the classic contrastive learning methods, a binary classification problem is developed that learns a set of discriminative hyperplanes – as a subspace – that will separate the two bags from each other in a weakly supervised manner. This subspace is then used as a descriptor for the video, dubbed discriminative subspace pooling. As the perturbed features belong to data classes that are likely to be confused with the original features, the discriminative subspace will characterize parts of the feature space that are more representative of the original data, and thus may provide robust video representations. To learn such descriptors, a subspace learning objective is formulated on the Stiefel manifold with Riemannian optimization methods for solving it efficiently.

In Chapter 6, novel objectives for one-class learning are explored, which we collectively refer to as Generalized One-class Discriminative Subspaces (GODS). The key idea is to learn a pair of complementary classifiers to flexibly bound the one-class data distribution, where the data belongs to the positive half-space of one of the classifiers in the complementary pair and to the negative half-space of the other. To avoid redundancy while allowing non-linearity in the classifier decision surfaces, each classifier is designed as an orthonormal frame and seek to learn these frames via jointly optimizing for two conflicting objectives, namely: i) to minimize the distance between the two frames, and ii) to maximize the margin between the frames and the data. The learned orthonormal frames will thus characterize a piecewise

linear decision surface that allows for efficient inference, while the objectives seek to bound the data within a minimal volume that maximizes the decision margin, thereby robustly capturing the data distribution. Several variants of proposed formulation are explored under different constraints on the constituent classifiers, including kernelized feature maps. The empirical benefits of this approach are demonstrated via experiments on data from several applications in computer vision, such as anomaly detection in video sequences, human poses, and human activities.

In Chapter 7, a conclusion and future work of this thesis is provided by summarizing proposed algorithm and contributions.

Chapter 2

Background

Video representation learning is a challenging research topic in machine learning, which has a wide range of applications. In this chapter, we will review the background of this topic and related mathematics. First, we will go through the popular model of convolutional neural networks, which has made considerable progress in a variety of fields. After that, we will follow the history to review the methodology of video representation learning in the recent decades. This includes traditional algorithm such as low-level hand-crafted feature and the state-of-the-art deep features from convolutional neural networks. In the meantime, we also review important works that aim to capture the temporal dynamics, such as RNNs, LSTM and GRU. At last, we summarize the pooling strategies that are used in the video representation learning and propose the research direction of this thesis.

2.1 Convolutional Neural Networks

Recently, the Convolutional Neural Networks (CNNs), one artificial neural network, has become dominant in various computer vision tasks, achieving state-of-the-art performances across a variety of domains [106, 170, 85, 73, 188, 142]. Video representation learning also benefit from the CNNs. Most of proposed algorithms in this thesis use the CNNs as the deep feature extractor. As a result, it will be briefly introduced.

The convolutional neural networks is composed of multiple building blocks, such as convolution layers, pooling layers, and fully connected layers, and is designed to automatically and adaptively process data that has a grid-like topology, such as an image. The convolution and pooling layers, perform feature extraction, whereas the fully-connected layer maps the extracted features into final predictions. This multiple-layer architecture enables it to represent an image from a combination of low-level features to highly-abstract semantic information. At the meantime, through a backpropagation algorithm, it optimizes

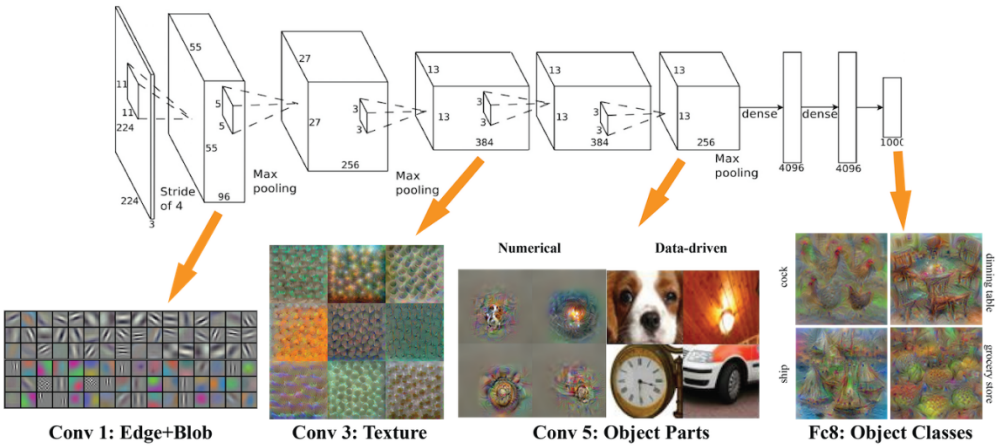


Fig. 2.1 An illustration of a typical CNNs and the feature visualization from its intermediate layers [106].

parameters in each layer, which minimizes the difference between CNN outputs and ground truth labels. Figure 2.1 demonstrates a typical modern CNN and a visualization of features from its intermediate layers.

The convolution layer is one of the most important building blocks of CNNs, which carries the main computation of the network. This layer performs dot product between two matrices, where one matrix is a group of learnable parameters, named as convolutional kernel, and the other matrix is an image patch from the original input. When the input extend its time dimension from image to video clip, the image patch will also be extended to a video block that has temporal dimension. The convolutional kernel will spatially (and temporally in 3D convolution) go through the entire input to produce an image representation namely as activation map, which gives the response of the kernel at each spatial position of the image. The sliding size of the kernel is called a stride. The size of activation map not only relies on the input size, but also the size of kernel, stride size and padding size if it's available. The relation between them can be written as:

$$W_{output} = \frac{W_{input} - K + 2P}{S} + 1, \tag{2.1}$$

where the W is the spatial dimension of the input and output, K is the kernel size, P is the padding size and S is the stride size.

The great success of the CNNs come from that it can learn highly discriminative, yet invariant feature representations from big data, which is the result of two key features of the CNNs: sparse interaction and parameter sharing.

Sparse interaction: Multi-layer perceptron (MLP) use the matrix multiplication connecting each output unit with every input unit, which is expensive. However, convolutional neural networks have the sparse interaction, which allow to detect meaningful features from image by using kernels with strides that have less computational cost. In other words, the image that has thousands or millions of pixels will be casted into tens or hundreds when it is processed by convolutional kernel with stride size.
Sharing of weights: Another difference between CNNs and multi-layer perceptron (MLP) is the sharing of weights. Weight matrices in MLP are used once and rarely revisited. However, a set of convolution kernels in each layer are forced to share a common value. In the convolutional layer, image patches from inputs will be applied the same weight to produce the feature map, which controls the complexity of the model.
Spatial invariance: In addition, CNNs are designed to be spatially invariant, that is they are not sensitive to the position of object in the image. This is achieved by applying the same filter bank to input patches at all locations, allowing CNNs to detect feature or objects even if it does not look exactly like the image.
Receptive field: At last, CNNs can easily have large receptive field with relatively less cost. The receptive field is the region in the input space that a particular CNN's feature is affected by. A larger receptive field is crucial to capture information about large objects. CNNs is able to increase the receptive field by stacking more convolutional layers.

Video data is consisted of images. When applied 2D CNNs directly on the video data frame by frame, it is found that CNNs could also achieve superior performance by leveraging the spatial information only [170]. However, to achieve better performance, video representations not only should understand the content in each frame, but also need to capture the motion dynamics associated with the temporal evolution. To this end, novel video representation learning schemes are required, as the image-based convolution kernel is hard to capture the temporal evolution in the video sequence. In the next section, we will review the background of video feature learning.

2.2 Video Feature Learning

The video feature learning is the process of learning how to extract discriminative information from video sequences as its representation. Discriminative information is generally task-dependent, emphasising motion dynamics in some situations while capturing temporal relationships in others. The most common computer vision task for video data is action recognition, which classify the video sequence into one or more action labels. Besides, video

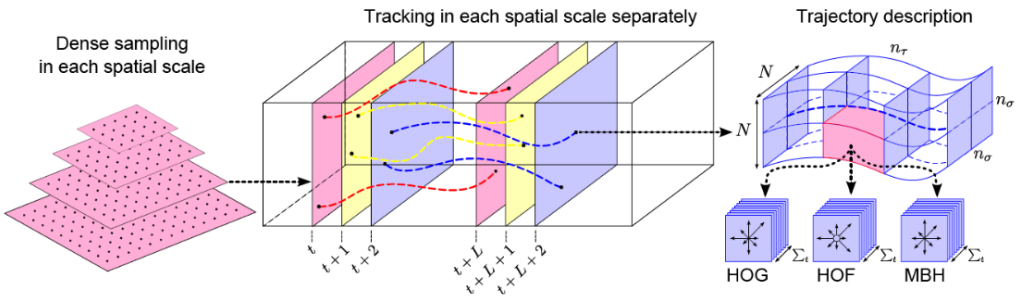


Fig. 2.2 An illustration of how to calculate HOG, HOF and MBH [200].

feature learning is also essential for action detection, action forecasting, video generation, and vision & language tasks such as video captioning, video Q & A, video dialogue.

2.2.1 Low-level hand-crafted video features

Traditional methods for video action recognition typically use hand-crafted local features, such as dense trajectories, HOG, HOF and MBH. [198], which model videos by combining dense sampling with feature tracking. In Figure 2.2 shows an illustration of how to calculate hand-crafted features. However, the camera motion, as one of the video natures, usually result in non-static video background and hurt the quality of features. To tackle this problem, Wang et al. [200] improved the performance of dense trajectories by removing background trajectories and warping optical flow. Based on the improved dense trajectories, high-level representations are designed via pooling appearance and flow features along these trajectories, and have been found to be useful to capture human actions. For example, Sadanand et al. [158] propose Action Bank, which converts the individual action detector into semantic and viewpoint space. Similarly, Bag of words model [172], Fisher vector [146], and VLAD [95] representations are mid-level representations built on such hand-crafted features with the aim of summarizing local descriptors into a single vector representation. In Peng et al. [144], a detailed survey of these ideas is presented. In many other works [60, 61, 82, 104, 170, 205, 24], the low-level hand crafted feature are also treated as an add-on to the CNN deep features. Experimental result indicate that the hand crafted feature is complementary to the deep features and further improve the performance. With the advancement of video benchmarks, it has been seen that the size of current benchmarks is growing. For example, the HMDB51 [107] and UCF101 [178] datasets were proposed in 2011 and 2012, which contained thousand of videos. whereas, the Kinetics-400/600/700 dataset [23, 22] and Howto100M dataset [130] are already in millions' scale, which are generated from 2018 to

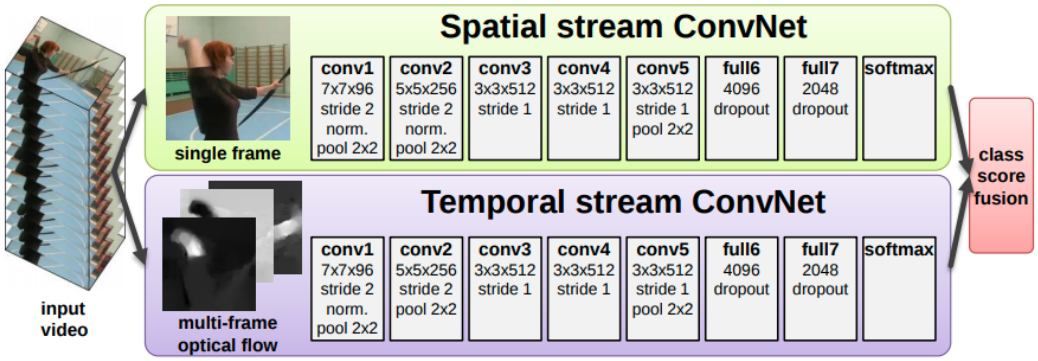


Fig. 2.3 Examples of two-stream architecture for video classification. [170]

2019. As a result of the high computational cost, low-level handcrafted features are currently seldom employed in video feature learning.

2.2.2 Video feature learning in CNNs

With the resurgence of deep learning methods for object recognition [106], there have been several attempts to adapt these models to video feature learning. Recent practice is to feed the video modalities, including but not limited to RGB frames, optical flow subsequences, 3D skeleton data and even natural language data into a deep (recurrent) network to train it in a supervised manner. Below, we will go through several video feature learning schemes in deep convolutional neural networks.

Two-Stream Architecture

One of the most successful methods is the two-stream models and its extensions [60, 61, 82, 104, 170], which is a direct extension from the image domain. As apparent from its name, it has two streams, spatial stream is to capture the appearance information from RGB frames and temporal stream is to learn the motion dynamics from stacked optical flow. The architecture of the two-stream networks can be seen in the Figure 2.3, which applies two independent CNNs to learn spatial and temporal video feature. Each frame and stacked optical flow images share the class label from their source video, and two streams are trained separately in supervised fashion by using the cross entropy loss. During the inference, to get the video-level prediction, they apply late fusion strategy to merge the prediction from multiple short clips.

As two streams are trained separately, they cannot complement each other before the late fusion. Based on the regular two-stream architecture, Feichtenhofer [60] propose new fusion strategies to enable the joint training of spatial and temporal streams. An illustration

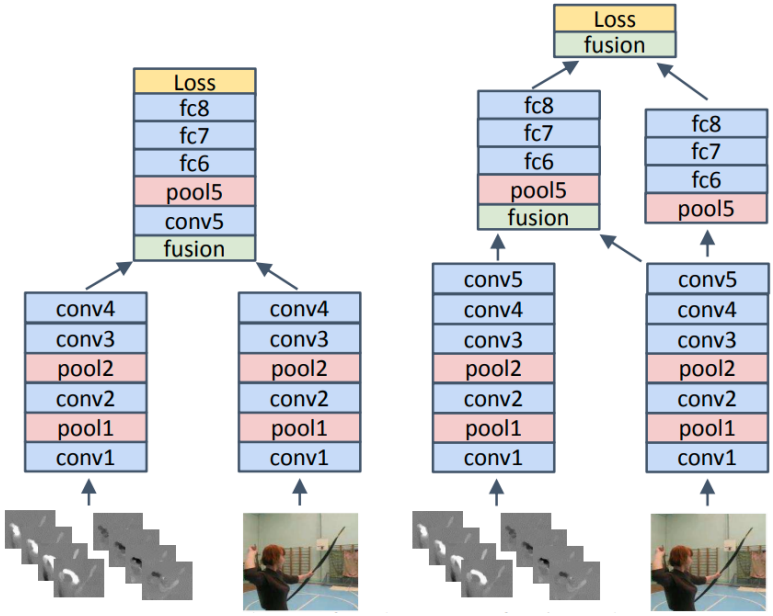


Fig. 2.4 Two examples of fusion strategy in the two stream networks. [60]

of two fusion examples is shown in the Figure 2.4. This architecture does not introduce new parameters compared to the previous methods, but provide a solution for incorporating the appearance and motion information of the video sequence and thus improve the performance for video classification.

Another innovation for two-stream stream is called Temporal Residual Networks [61], which apply the residual connection between each time step for capturing both spatial and temporal video features. Similarly, Wang et al. [211] propose two-stream based temporal segment networks, which models the entire video sequence. Before using two-stream networks, it splits the video input into numerous short clips at various time steps. To reach the final forecast, it computes the segment consensus. Although the architectures of these networks differ, the basic idea is to split the video into short clips and embed them in a semantic feature space, then recognise the actions either by aggregating the individual features per clip using a statistic (such as max or average) or by directly training a CNN-based end-to-end classifier [60]. While the latter schemes are appealing, they usually need to store the feature maps from all the frames in memory which may be prohibitive for longer sequences. Moreover, this training strategy may fail to capture the long-term dynamics in the video sequence, because the optical flow only captures short-term motions and it is impractical to feed large amount of flow streams into the neural network.

Recurrent Neural Networks and Long Short Term Memory.

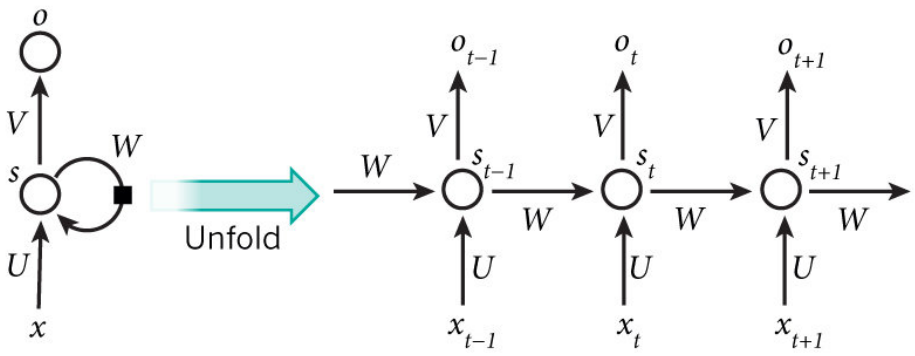


Fig. 2.5 An illustration of an unfolded Recurrent Neural Network.

Recurrent neural networks (RNNs) have been an interesting and important part of neural network research since the 1990's [215]. It has already been applied to a wide variety of problems involving time sequences of events and ordered data such as characters in words and frames in video sequences. RNNs take a sequence of data into its recurrent architecture and sequentially produce output at each time step. RNNs are consist of several RNN units and each unit has a hidden state. The current time step's hidden state is calculated using information from the previous time step's hidden state and the current input, allowing the model to retain knowledge from the previous time step when processing the current time step's data, thus capturing the temporal dependencies. A detailed pipeline in demonstrated in the Figure 2.5.

Specifically, an input sequence can be denoted as $(x^1, x^2, ..., x^T)$, in which the $x^t \in \mathbb{R}^D$ is the input data at time step t. The input from each time step will be connected with the hidden state via a weight matrix U, and the hidden state of current time step, s_t, will be passed to the next time step through another weight matrix W. At last, the output, o_t from each time step combine the hidden state from previous time step and the input from the current time step via a weight matrix V. Formally, the formulation of RNN can be written as:

$$o_t = \sigma_o(Vs_t + b_o) \tag{2.2}$$

where s_t is:

$$s_t = \sigma_s(Ws_{t-1} + Ux_t + b_s) \tag{2.3}$$

In the equation above, the σ and b are the non-linear activation function and bias of the output or hidden layers.

It can be seen from the equation that each time step's result is the sum of the current input and all prior history. As a result, depending on the previous context, this aggregated data may generate predictions.

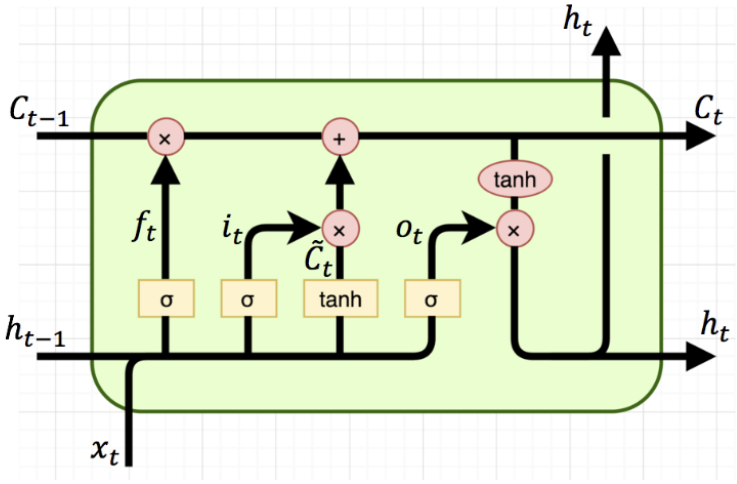


Fig. 2.6 An illustration of a LSTM cell.

However, such simple RNN architecture fails to capture long-term dependencies. When the time span grows longer, RNNs are unable to retain the information that has a longer history. This is due to the fact that the gradient of the loss function decays exponentially with time [143]. If gradient values are close to zero or greater than one, the gradient from longer past is easily to be vanished or exploded with the accumulation of time, and thus, the network ignores the long-term dependencies and hardly connect the inputs from temporally distant steps. This phenomenon is named as gradient vanishing and gradient exploding [10, 90], which are regarded as fundamental limitations of the RNN architecture.

To tackle this problem, researchers start to utilize logistic gates and hidden states in the RNN architecture. Works such as long-short term memory (LSTM) [91] and gate recurrent unit (GRU) [38] are proposed. GRU is a simplified version of LSTM. let's take LSTM as example, it introduces memory cells, in which, the input and output are controlled by different logistic gates. Different from the traditional RNN unit, these gates are able to add or remove information to the cell state and maintain information in memory for long periods of time. Moreover, the cell state is designed as a conveyor belt. It runs straight down to the end of the chain, with only some minor linear interactions, encouraging information as well as gradient just flow along it unchanged. Figure 2.6 demonstrates a typical architecture of LSTM cell. In which, $C, h, x, f, i, \tilde{C}, o$ are cell state, hidden state, input, forget gate, input gate, new candidate value and output gate. Formally, the LSTM calculation can be written as following:

$$i_t = \sigma(x_t U^i + h_{t-1} W^i) \tag{2.4}$$
$$f_t = \sigma(x_t U^f + h_{t-1} W^f) \tag{2.5}$$
$$o_t = \sigma(x_t U^o + h_{t-1} W^o) \tag{2.6}$$
$$\tilde{C}_t = tanh(x_t U^g + h_{t-1} W^g) \tag{2.7}$$
$$C_t = \sigma(f_t * C_{t-1} + i_t * \tilde{C}) \tag{2.8}$$
$$h_t = tanh(C_t) * o_t \tag{2.9}$$

where σ and tanh are activation function of sigmoid and hyperbolic tangent respectively.

From above formulation, it can be seen that the forgot gate produce 0 or 1 to the previous cell state, which control whether or not to forgot the information from previous time step. The nest step is to decide what new information that are going to store in the cell state. First, the input gate decides which values are going to be updated. Next, a tanh layer creates a vector of new candidate values, $\tilde{C}$, that could be added to the state. After removing and adding information, we will have the new cell state. Finally, the output gate decides which parts of the cell state is needed. Then, the cell state will go through tanh (to push the values to be between -1 and 1) and multiply it by the output of the sigmoid gate, so that only the desired part is filtered.

When it applies to the video feature learning, the input information will be the frame-level feature from the deep neural network and recurrent neural networks will learn to capture the temporal dynamic inside the video sequence. Based on this, some works that utilize recurrent models in video feature learning are proposed [5, 50, 53, 115, 179, 226].

3D Convolutional Neural Networks.

Another promising solution is to use 3D convolutional filters [24, 188, 213, 233, 208]. As a direct extension of 2D convolutional kernels, 3D kernel expand along with the time dimension. Thus, 3D filters can capture both spatial and temporal video structure from the video sequence. Instead of conducting convolution in the image domain along with the vertical and horizontal axes, 3D convolution will include an extra depth axis. However, feeding the entire video into the 3D CNNs may be computationally prohibitive. To reduce the computational cost, [24] proposed to use inflated 2D kernels to bootstrap the 3D kernel, which require less cost for learning the 3D kernel from scratch. On the other side, [189] propose R(2+1)D convolutional block, which decouples the spatial and temporal convolution while significantly gains in accuracy. Figure 2.7 demonstrates the difference of R(2+1)D convolutional kernel and regular 3D convolutional kernel. However, 3D or (2+1)D kernels

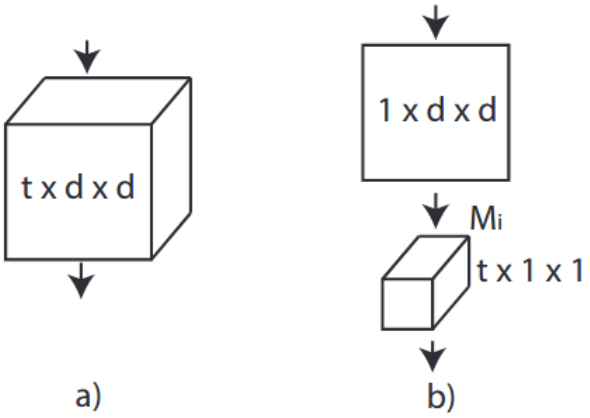


Fig. 2.7 a) the regular 3D convolutional kernel in the size of t x d x d, where t denotes the temporal extend and d denotes the width and height.b) the R(2+1)D convolutional kernel, which decouple the 3D kernel into a spatial 2D convolution followed by a temporal 1D convolution.

bring more parameters into the architecture; as a result, may demand large and clean data for effective training [24]. To this end, [213] present the non-local neural network, which introduce the attention mechanism to capture the non-local when doing the convolution. From those previous works, we believe that an effective CNN architecture that can extract useful action-related features which is essential to make progress in video understanding.

Video Transformers.

More recently, the Transformer model [195] has received much attention due to its impressive performance in the field of natural language processing (NLP) [49]. While CNNs rely on the local operation of convolution, the building block of transformers is self-attention [195] which is particularly effective at modelling long-range dependencies. In the image domain, the Vision Transformer (ViT) [52] was proposed as a convolution-free architecture which uses self-attention between non-overlapping patches in all layers of the model. In Figure 2.8, a standard QKV attention is demonstrated, which generate query (Q), key (K) and value (V) for each image patch. Each query will compute similarity with keys and formulate attention weightings and the final output is the weighted sum between values and attention weightings., which will then be used to update the query patch. ViT was shown to be competitive with state-of-the-art CNNs on the task of image categorization. In the last few months, several adaptations of ViT to video have been proposed [11, 135, 4, 207]. In the video domain, transformer architecture has its huge advantage. Each image patch will be treated equally in the self-attention architecture, which enables the model to build both local and global dependencies from the first layer to the last layer., which is beneficial for the video representation learning. Unlike recurrent network, it will not face the gradient vanish issue.

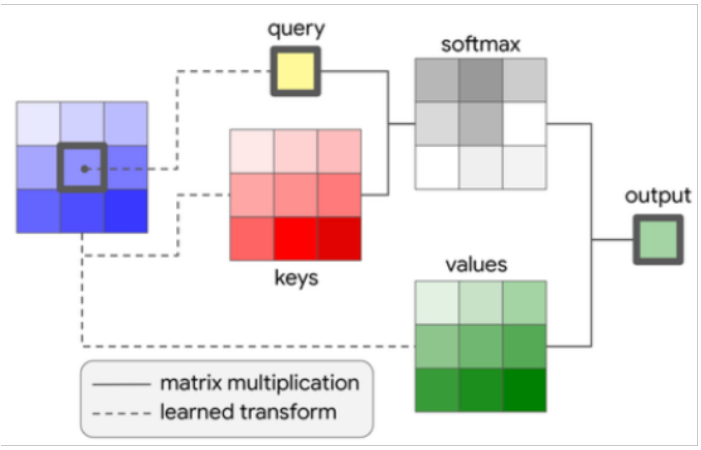


Fig. 2.8 Illustration of standard QKV self-attention block used in the vision transformer. The final output is the weighted sum between values and attention weighting.

Unlike CNNs, it does not need to wait until the later stage for building global relationship.In order to capture salient temporal information from the video, these works typically extend the self-attention mechanism to operate along the time axis in addition to within each frame.

Since video transformers have a larger numbers of parameters and fewer inductive biases compared to CNNs, they typically require large-scale pretraining on supervised image datasets, such as ImageNet-21K [153] or JFT [4], in order to achieve top performance. In this thesis, the method is still based on CNNs architecture for saving computational cost. However, this thesis adopts similar motivation with the self-attention block used in video transformers without prior consultation, assigning different weighting for each input instance, which will be introduced in the rest of this thesis.

2.3 Pooling strategy in video data

One nature of the video data is its redundancy. Compared to the image data, video sequences include more than one image, which increase the computational cost as well as the difficulties to extract meaningful features. Thus, people apply various pooling schemes spatially and temporally to aggregate video data while reducing computational cost and improving spatial invariance. Typically, pooling schemes consolidate input data into compact representations based on some data statistic that summarizes the useful content. For example, average and max pooling captures zero-th and first order statistics, which are the most common pooling scheme in the deep neural network. There are also works that use higher-order pooling, such as Cherian and Gould [32] using second-order, Cherian et al. [34] using third-order, and Girdhar et al., [70] proposing a video variant of the VLAD encoding which is approximately

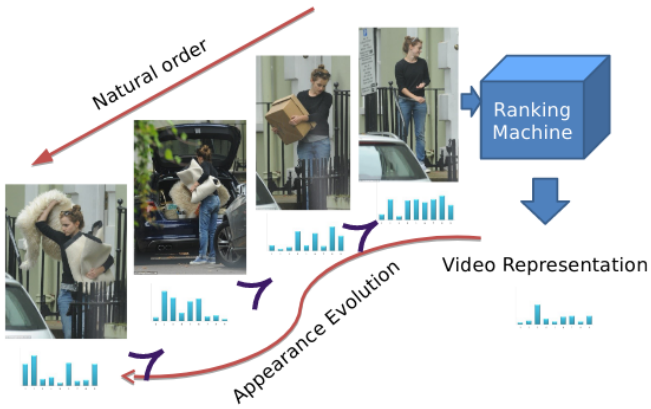


Fig. 2.9 An illustration of how rank pooling work. The ranking function is able to capture the evolution of the appearance over time from the video sequence. Its parameters is used as video representation, which embed the motion information. [64]

a mixture model. A recent trend in pooling schemes, which we also follow in this thesis, is to use the parameters of a data modelling function, as the representation. For example, rank pooling [64] is to use the parameters of a support vector regressor as a video representation. A detailed illustration of how rank pooling works is shown in the Figure 2.9. In Bilen et al., [13], rank pooling is extended towards an early frame-level fusion, dubbed dynamic images, which are used to train the CNNs afterwards; Cherian et al. [31] generalized rank pooling to include multiple hyperplanes as a subspace, enabling a richer characterization of the spatio-temporal details of the video. This idea was further extended to non-linear feature representations via kernelized rank pooling in [35]. In this thesis, the focus is also on designing "smart" pooling schemes which meet the characteristic of video data and improve the quality of video representations. In the Chapter 3, we following the rank pooling idea to generate the dynamic flow images. In Chapter 4 and 5, we improve the formulation and objective to learn more discriminative video representation, which use the parameters of a binary classifier to be the video level descriptor. Specifically, it is trained to classify the frame level features from a another bag of negative features. Finally, in the Chapter 6, we investigate a variant of our pooling scheme for the one-class problem, which demonstrates the applicability of our algorithm in different problem set-up. As far as we have witnessed, we are the first to use the parameters of a binary classifier to be the video level descriptor. The most recent popular work, contrastive learning [84, 28, 75], share the similar motivation with the Chapter 4 and 5, but the problem set-up and objective are completely different.

2.4 Datasets

2.4.1 Datasets

To evaluate proposed methods in this thesis, we experiment them in various benchmarks. In the following, benchmarks used in this thesis will be introduced.

HMDB-51 [107] and UCF-101 [178]: are two popular benchmarks for video action recognition. Both datasets consist of trimmed videos downloaded from the Internet. HMDB-51 has 51 action classes and 6766 videos, while UCF-101 has 101 classes and 13320 videos. Both datasets are evaluated using 3-fold cross-validation and mean classification accuracy is reported.

Charades [168]: is an untrimmed and multi-action dataset, containing 11,848 videos split into 7985 for training, 1863 for validation, and 2,000 for testing. It has 157 action categories, with several fine-grained categories. In the classification task, we follow the evaluation protocol of [168], using the output probability of the classifier to be the score of the sequence. In the detection task, we follow the 'post-processing' protocol described in [167], which uses the averaged prediction score of a small temporal window around each temporal pivot. In this thesis, we evaluate the performance on both tasks using mean average precision (mAP) on the validation set. by using the provided two-stream fc7 feature[1].

Kinetics-600 [101]: is one of the largest dataset for action recognition. It consists of 500K trimmed video clips over 600 action classes with at least 600 video clips in each class. Each video clip is at least 10 seconds long with a single action class label.

MPII Cooking Activities Dataset [154]: consists of high-resolution videos captured by a static camera, showing 14 different people cooking various dishes with 64 distinct activities. There are 3,748 video clips and 1,861 clips for the background. This dataset has several actions that are subtle, such as 'peeling', 'cutting in', 'cutting apart', etc., and are highly imbalanced with regard to the number of videos per action.

MSR Action3D [119] and NTU-RGBD [165]: are two popular action datasets providing 3D skeleton data. Specifically, MSR Action3D has 567 short sequences with 10 subjects and 20 actions, while NTU-RGBD has 56,000 videos and 60 actions performed by 40 people from 80 different view points. NTU-RGBD is by far the largest public dataset for depth-based action recognition. The videos have on average 70 frames and consist of people performing various actions; each frame annotated for 25 3D human skeletal keypoints (some videos have multiple people). According to different subjects and camera views, two evaluation protocols are used, namely cross-view and cross-subject evaluation [165].

[1] http://vuchallenge.org/charades.html

Public Figures Face Database (PubFig) [108]: contains 60,000 real-life images of 200 people. All the images are collected directly from the Internet without any post-processing, which make the images in each fold have large variations in lighting, backgrounds, and camera views. Unlike video-based datasets, PubFig images are non-sequential.
YUP++ dataset [61]: is a recent dataset for dynamic video-texture understanding. It has 20 scene classes with 60 videos in each class. Importantly, half of the sequences in each class are collected by a static camera and the rest are recorded by a moving camera. The latter is divided into two sub-datasets, YUP++ stationary and YUP++ moving. As described in the [61], we apply the same 1/9 train-test ratio for evaluation. There are about 100-150 frames per sequence.
JHMDB dataset: is a video action recognition dataset [96] consisting of 968 clips with 21 classes, which is a subset of HMDB dataset [107].
UCF-Crime dataset: is the largest publicly available real-world anomaly detection dataset [180], consisting of 1900 surveillance videos and 13 crime categories such as *fighting*, *robbery*, etc. and several "normal" activities, such as the daily walking, running and driving. Illustrative video frames from this dataset and their class labels are shown in Figure 6.4.
UCSD Ped2 dataset: contains 16 videos in the training and 12 videos in the test set. There are 12 abnormal events in the test videos, such as the *Biker*, *Cart*, *Skater*, etc.

2.5 Chapter Summary

In this Chapter, we have the literature review relating to this thesis. We began by explaining convolutional neural networks and how it works, as well as why it could achieve superior performance in many computer vision tasks. Then, in comparison to image feature learning, we show how difficult it is to learn video feature representation. Further, we survey the related works of video feature learning, which includes the traditional low-level hand crafted video feature learning and the video feature learning algorithm in CNNs and video transformers. Capturing the temporal evolution or motion dynamic within the video series is determined to be one of the most significant approaches in video feature learning. Furthermore, the repetition of the video sequence is another feature that makes it challenging to apply image-based algorithms to video. Finally, we introduced benchmarks used in this thesis and discussed the most common pooling strategies in video data and proposed the motivation for this thesis, which is to design a pooling strategy that aids in the aggregate of frame-level features into video-level representation while capturing temporal evolution and discriminative semantic information. The literature that are highly related with each chapter will be separately reviewed in each chapter.

Chapter 3

Ordered Pooling of Optical Flow Sequences for Action Recognition

Training of Convolutional Neural Networks (CNNs) on long video sequences is computationally expensive due to the substantial memory requirements and the massive number of parameters that deep architectures demand. Usually, deep architectures for video based action recognition take as input short video clips consisting of one to a few tens of frames. Using longer sub-sequences would require deeper networks or involve a huge number of parameters, which might not fit in the GPU memory, or may be problematic to train due to computational complexity. This restriction and thus clipping of the temporal receptive fields of the videos to short durations prohibit CNNs from learning long-term temporal evolution of the actions, which is very important in recognition especially when the actions are complex. One standard way to tackle this difficulty in capturing long-term dependencies is to use temporal pooling that can be applied either when providing the input to the CNN or after extracting features from intermediate CNN layers [13].

In this chapter, we explore a recently introduced early fusion scheme based on a Ranking SVM [64] formulation where several consecutive RGB frames in the video are fused to generate one "dynamic image" by minimizing a quadratic objective with temporal order constraints. The main idea of this scheme is that the parameters learned after solving this formulation capture the temporal ordering of the pixels from frame-to-frame, thus summarizing the underlying action dynamics. One drawback of the rank pooling approach is that the formulation does not directly capture the motion associated with the action – it only captures how the pixel RGB values change from frame-to-frame. Typically, the video pixels can be noisy. Given that the pooling constraints in this scheme only look at increasing pixel intensities from frame-to-frame, it can fit to noise pixels that adhere to this order, however,

are unrelated to the action. To mitigate these issues, in this thesis we look at the rank pooling formulation in the context of optical flow images instead of RGB frames.

It is clear that optical flow can easily circumvent the above problems associated with sequences of RGB frames. The flow by itself captures the objects in motion in the videos, and thereby capturing the action dynamics directly, while thresholding the flow vectors helps to avoid noise. Thus, we posit that summarizing sequences of optical flow can be significantly more beneficial than using RGB frames for recognizing motion-related categories. By solving the rank-SVM formulation (Section 3.2), we generate flow images that summarize the action dynamics, dubbed *dynamic flow images*. These images are then used as input to a standard action recognition CNN architecture to train for the actions in the sequences. In Figure 3.1, we show a few sample dynamic flow images generated using the proposed technique for the respective RGB frames.

A natural question here can be regarding the intuitive benefit of using such a flow summarization scheme, given that standard action recognition CNN frameworks already use a stack of flow frames [170]. Note that such flow stacks usually use only a few frames (usually 10), while using the dynamic flow images, we summarize several tens of flow frames, thereby capturing long-term temporal context. However, the precision of optical flow can be easily affected by the dynamic camera and the complex motion distribution can be barely captured by optical flow. Through extensive experiments in this chapter, we observe improvement in all benchmarks by including optical flows. But still, more challenging benchmarks with moving background and complex motions should be explored in the future work, which will be discussed in the last chapter.

To validate our claims, we provide extensive experimental comparisons (Section 3.3) on two standard benchmark action recognition datasets, namely (i) the HMDB-51 dataset, and (ii) the UCF-101 dataset. Our results show that using dynamic flow images lead to significant improvements in action recognition performance. We find that this leads to 4% improvement on HMDB-51[107] and 6% on the UCF-101[178] dataset in comparison to using dynamic RGB images without combining any other methods.

3.1 Related work

3.1.1 Dynamic Image

The rank pooling is also named as temporal pooling, which is to model the video-wide temporal evolution of appearance in the video. The concept of it is first introduced by Fernando et al. [64]. In their paper, they propose to capture the temporal ordering of a video by training

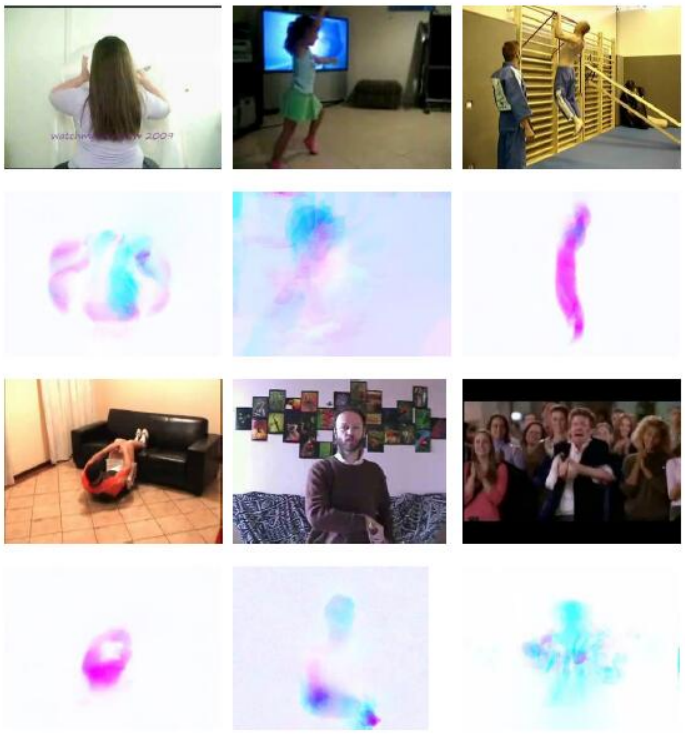


Fig. 3.1 Examples of dynamic flow images.

a linear ranking machine, which project all frames of a video onto a hyper-plane and enforce the projection to follow the chronological order. And then, the parameters of the linear ranking function is used to be the representation of the video. Following by this, a couple of extension works are demonstrated, such as the hierarchical rank pooling Fernando et al. [63], which divide a video into several clips and apply rank pooling hierarchically to get a higher order representation. In addition, Fernando and Gould [65] introduce a temporal pooling layer, which can sit above any CNN architecture and be end-to-end trainable for all parameters of the model.

In the rank pooling, the parameter of the ranking machine is used as the video level representation, as it encode the temporal evolution of the video appearance. Apart from this, the dimension of the rank pooled feature will keep the same as its input. By using this property, Bilen et al. [13] introduce the dynamic image, which applies rank pooling on the top of RGB frames from one video and reshape the rank pooled feature back as an image, namely dynamic image. Based on this, a few extension works are proposed afterwards, such as the GRP [31] that demonstrate a generalized version of rank pooling and [66] which embed the dynamic image in an end-to-end manner. In the second row of Figure 3.2, we show a few examples of dynamic images. Specifically, the dynamic image can be expressed

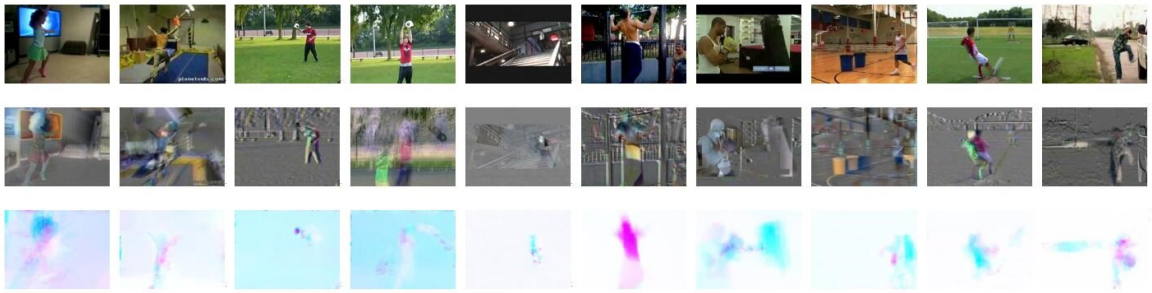


Fig. 3.2 Examples of original RGB images (the first row), dynamic image (the second row) and dynamic flow images (the third row).

in the form of RankSVM [175]:

$$w^* = \underset{w\in\mathbb{R}^p,\zeta\geq 0}{\arg\min}\ P(w) := \frac{1}{2}\|w\|^2 + C\sum_{i<j}\zeta_{ij} \tag{3.1}$$
$$\text{subject to } \langle w, I_j\rangle - \langle w, I_i\rangle \geq 1-\zeta_{ij}, \quad \forall i<j$$

where $\langle .,. \rangle$ is the inner product operation, $I_1, I_2 ... I_n$ is the RGB frames of one video, ζ is a slack variable, C is the penalty parameter, and w is the decision boundary of the RankSVM, which is reshaped as dynamic image. From the formulation, the decision boundary capture the temporal evolution of the image appearance. However, the formulation does not directly capture the motion associated with the action – it only captures how the pixel RGB values change from frame-to-frame. Typically, the video pixels can be noisy and the pooling constraints only look at the increasing pixel intensities from frame to frame. As a result, the noise pixels that adhere to this order could also be captured in the dynamic images.

3.2 Approach

To mitigate these issues, we look at the rank pooling formulation in the context of optical flow images instead of RGB frames. In the Figure 3.2, we show a few sample dynamic flow images (the third row) generated using the proposed technique for the respective RGB frames. Compared with the dynamic images (the second row), the dynamic flow images focus more on the motion pattern.

3.2.1 Dynamic Flow Image

Let us assume that we are provided with a sequence of n consecutive optical flow images $\mathscr{F} = [f_1, f_2, ..., f_n]$, where each $f_i \in \mathbb{R}^{d_1\times d_2\times 2}$, where d_1, d_2 are the height and width of the

image. We assume a two-channel flow image corresponding to the horizontal and vertical components of the flow vector, which we denote by f_i^u and f_i^v respectively. Our goal is to generate a single "dynamic flow" image $F \in \mathbb{R}^{d_1 \times d_2 \times 2}$ that captures the temporal order of the flow images in $\mathscr{F}$. We use the following formulation to obtain this representation.

$$\begin{aligned} \min_{F \in \mathbb{R}^{d_1 \times d_2 \times 2}, \xi \geq 0} \quad & \|F\|^2 + C \sum_{i<j} \xi_{ij} \\ \text{s. t. } & \langle F, f_i \rangle \leq \langle F, f_j \rangle - 1 + \xi_{ij}, \quad \forall i < j, \end{aligned} \tag{3.2}$$

where $\langle ., . \rangle$ represents an inner product between the optical flow vectors and the dynamic flow image to be found. As one may recall, this formulation is similar to the Ranking-SVM formulation [98], where the inner product captures the ranking order, which in our case corresponds to the temporal order of the frames in the sequence. While, ideally, we enforce that the projection (via the inner-product) of the flow frames to the dynamic flow image is lower bounded by one, we relax this constraint using the variables ξ with the penalty term C, as is usually done in a max-margin framework. While, the above formulation is a direct adaptation of the Ranking SVM formulation, there are a few subtleties that need to be taken care when using this formulation for optical flow images. We will discuss these issues next.

It is often observed that a direct use of raw optical flow in (3.2) is often inadequate. This is because computing optical flow is a computationally expensive and difficult task, and the flow solvers are often prone to local minima, leading to inaccurate flow estimates. In order to circumvent such issues, we instead apply the algorithm on flow images computed over running averages; such a scheme averages the noise in the flow estimates. For the flow set $\mathscr{F}$, we represent the resulting smoothed flow image for frame f_t as:

$$\hat{f}_t = \frac{1}{t} \sum_{i=1}^{t} f_i. \tag{3.3}$$

We compute the dynamic flow on such smoothed flow images. Note that since flow is signed, such averaging will cancel white noise.

Another issue specific to optical flow images is that the two flow channels u and v are coupled. They are no more the color channels as in an RGB image, instead they represent velocity vectors, which together capture the motion vector at a pixel. However, we assume independence in the channels in Equation 3.2, which do not include dependencies between two channels. One way to avoid this difficulty is to decorrelate these channels via diagonalization; i.e., use the singular vectors associated with a short set of flow images as representatives of the original flow images. However, our experiments showed that this did not lead to significant improvements. Thus, we choose to ignore this coupling in the this

chapter, and assume each channel is independent when creating the dynamic flow images. Our experiments (in Section 3.3) show that this assumption is not detrimental.

Incorporating the above assumptions into the objective, and assuming $F_u, F_v \in \mathbb{R}^{d_1 \times d_2}$ are the two dynamic flow channels corresponding to the horizontal and vertical flow directions, we can rewrite (6.3) as:

$$\min_{F_u, F_v \in \mathbb{R}^{d_1 \times d_2}, \xi \geq 0} \|F_u\|^2 + \|F_v\|^2 + C \sum_{i<j} \xi_{ij}$$

subject to

$$\langle F_u, \hat{f}_i^u \rangle + \langle F_v, \hat{f}_i^v \rangle \leq \langle F_u, \hat{f}_j^u \rangle + \langle F_v, \hat{f}_j^v \rangle - 1 + \xi_{ij}, \forall i < j. \tag{3.4}$$

We solve the above optimization problem using the libSVM package as described in [13]. Once the two flow images F_u and F_v are created, they are stacked to generate a two channel dynamic flow image, which is then fed to a multi-stream CNN as described for learning actions (as described in the next section). As is clear from the Figure 3.2, the dynamic flow images summarizes the actual action dynamics in the sequences, while the dynamic RGB images include averaged background pixels and thus are may include dynamics unrelated to recognizing actions.

3.2.2 Three-Stream Prediction Framework

Next, we propose a three-stream CNN setup for action recognition. This architecture is an extension of the popular two-stream model [170] that takes as input individual RGB frames in one stream and a small stack of optical flow frames (about 10 flow images) in the other. One shortcoming of this model is that it cannot see long-range action evolution, for which we propose to use our dynamic flow images (that summarizes about 25 flow frames in one dynamic flow image). As is clear, each such stream looks at complementary action cues. Our overall framework is illustrated in Figure 3.3, which consists of an optical flow stream taking stacks of flow frames, an RGB stream taking single RGB frames, and our dynamic flow stream taking single images, each image summarizing the action dynamics over 25 flow frames.

To be precise, for the dynamic flow stream, for each video sequence, we generate multiple dynamic flow images. In order to achieve this, we first split the input flow video into several sub-sequences each of length w and generated at a temporal stride s. For each sub-sequence, we construct a dynamic flow image using the optical flow images in this window. We associate the same ground truth action label for all the sub-sequences, thus effectively increasing the number of training videos by a factor of $\frac{n}{s}$, where n is the average number of frames in

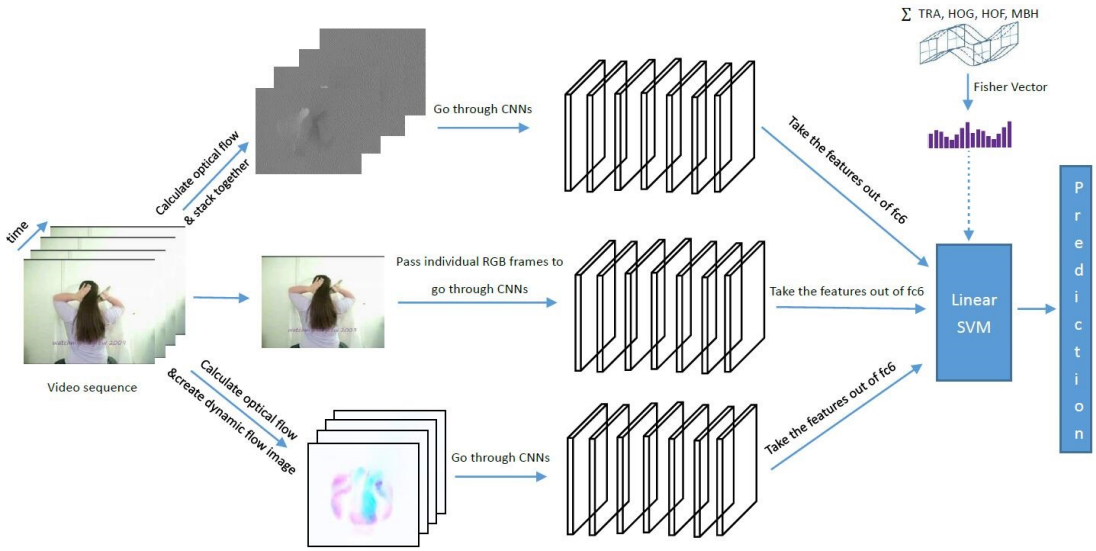


Fig. 3.3 Architecture of our dynamic-flow CNN based classification setup.

the sequences. Note that we use a separate CNN stream on dynamic flow images. Given that action recognition datasets are usually tiny, in comparison to image datasets (such as ImageNet), increasing the training set is usually necessary for the effective training of the network.

3.2.3 Practical Extensions

We use the TVL1 optical flow [227] algorithm to generate the flow images using its OpenCV implementation. For every flow image, we subtract the median flow, thus removing camera motion if any (assuming flow from the action dynamics occupies a small fraction with respect to the background). The resulting flow images are then thresholded in the range of $[-20, 20]$ pixels and setting every other flow vector to zero. This step thus removes unwanted flow vectors, that might correspond to noise in the images. Next, we scale the flow to the discrete range of $[0, 255]$, and convert each flow channel as a gray-scale image. A set of such transformed flow images are then used as input to the rank-SVM formulation in (3.3), thereby generating one dynamic flow image per sub-sequence. Using libSVM package, it takes about 0.25 seconds to generate one dynamic flow image on a single core CPU combining 25 flow frames each of size $224 \times 224 \times 2$.

3.3 Experiments

In this section, we first describe the datasets used in our experiments, followed by an exposition to the implementation details of our framework and comparisons to prior works.

3.3.1 Implementation Details

Training CNNs

In the experiments to follow, we use the two successful CNN architectures, namely AlexNet [106] and VGG-16 [171]. We use the Caffe toolbox [97] for the implementation. As the number of training videos is substantially limited to train a standard deep network from scratch, we decided to fine-tune the networks from models pre-trained for image recognition tasks (on the ImageNet dataset). Similar with the Dynamic Image Networks [12], we also use multiple dynamic flow and dynamic images per video. On the training subsets of our dataset, we fine-tune all the layers of the respective networks with an initial learning rate of 10^{-4} for VGG-16 and 10^{-3} for Alex net with a drop-out of 0.85 for 'fc7' and 0.9 for 'fc6' as recommended in [171, 62]. The drop-out is subsequently increased when the validation loss begins to increase. The network is trained using SGD with a momentum of 0.9 and weight decay of 0.0005. We use a mini-batch size of 64 for HMDB51 and 128 for UCF101.

Fusion Strategy

As alluded to earlier, we use a three-stream network with RGB, stacked-optical flow, and dynamic flow images. The three networks are trained separately. During testing, output of the fc6-layer of each stream is extracted (see Figure 3.3). These intermediate features are then concatenated into a single vector, which is then fused via a linear SVM. We also experimented with features from the fc7 layer, however we found them to perform slightly inferior compared to fc6. As is typically done, we also extract dense trajectory features from the videos (such as HOG, HOF, and MBH) [199], which are encoded using Fisher vectors, and are concatenated with the CNN features before passing to the linear SVM.

3.3.2 Experimental Results

We organize our experiments into various categories, namely (i) to decide the hyper-parameters of our formulation (e.g., the window size to generate dynamic flow images), (ii) benefits of using dynamic flow images against dynamic RGB images and other data modalities, (iii) influence of the CNN architecture, (iv) complementarity of the dynamic flow

Table 3.1 Influence of using different number of consecutive flow frames (window size) to construct one dynamic flow image. This experiment uses the split 1 of HMDB51 dataset with the VGG-16 CNN model.

Dynamic Flow window size	Accuracy
15	48.23%
25	**48.75%**
30	46.60%

images to hand-crafted features, and (v) comparisons against the state of the art. Below, we detail each of these experiments.

Hyper-parameters: We tested the performance of dynamic flow images using different temporal window sizes, i.e., the number of consecutive flow frames used for generating one dynamic flow image. The results of this experiment on the HMDB51 dataset split 1 is provided in Table 3.1. As is clear, increasing window sizes is not beneficial as it may lead to more action clutter. Motivated by this experiment, we use a window size of 25 at a temporal stride of 5 in all the experiments in the sequel.

Benefits over Dynamic Images

As discussed in Section 3.2, dynamic flow (DF) captures the action dynamics directly in comparison to the dynamic RGB images (DI) [13]. To demonstrate this, we show experiments using the VGG-16 network in Table 3.2 on the HMDB-51 dataset and using the Alexnet network on the UCF-101 dataset in Table 3.3[1]. We also show comparisons to RGB, stack of flow, and various combinations of them. As is clear from the two tables, DF + RGB is about 9-10% better than DI + RGB consistently on both the datasets and both CNN architectures. Surprisingly, combining DI with DF + RGB leads to a reduction in performance of about 4% (Table 3.2). This, we suspect, is because DI includes pixel dynamics from the scene background that may be unrelated to the actions and may confuse the subsequent classifier; such noise is avoided by computing optical flow. In order to explain this phenomenon, we create the Table 3.4 to display the the accuracy in each action in HMDB-51 split1 over three methods, which is able to fully compare the relationship among RGB, dynamic image and dynamic flow. From Table3.4, for some actions, such as 'cartwheel','run' and 'somersault', the performance of dynamic image + RGB always drag the performance of dynamic flow + RGB. The most confusing classes in the third column are always the same ones in the first column. As mentioned in Section3.1, we believe the dynamic image capture some non-

[1]The results of Alexnet on UCF-101 was taken directly from [13].

Table 3.2 Evaluation on HMDB51 split 1 using VGG-16 model.

Method	Accuracy
Static RGB[62]	47.06%
Stacked Optical Flow [62]	55.23%
Dynamic Image [13]	44.74%
Dynamic Flow	48.75%
Dynamic Image+RGB	47.96%
Dynamic Flow+RGB	**58.30%**
Dynamic Image+Dynamic Flow+RGB	54.86%
(S)Optical Flow+RGB[62]	58.17%
Dynamic Image+RGB+(S)Optical Flow	55.40%
Dynamic Flow+RGB+(S)Optical Flow	**61.70%**

Table 3.3 Evaluation on UCF101 using AlexNet CNN model.

Method	Accuracy
On split 1	
Static RGB[209]	71.20%
Stacked Optical Flow[209]	80.10%
Dynamic Flow	75.36%
Dynamic Flow + RGB	**84.93%**
(S)Optical Flow + RGB[209]	84.70%
Dynamic Flow + RGB + (S)Optical Flow	**88.63%**
Over three splits[2]	
Static RGB	70.10%
Dynamic Flow	76.19%
Dynamic Image[13]	70.90%
Dynamic Image + RGB[13]	76.90%
Dynamic Flow + RGB	**84.93%**

motion related information from the background. That is why the performance of dynamic image + RGB become even worse compared with the one of dynamic flow + RGB. After dynamic image combining with dynamic flow and RGB, the drawback of non-motion related information from RGB and dynamic image beat the advantage of the spatial information from RGB, and thus confuse the classifier when doing the prediction and then, accuracy drops.

Benefits of Three-stream Model

In Tables 3.2 and 3.3, we also evaluate our three-stream network against the original two-stream framework [171]. It can be seen that the performance of DF + RGB is similar to

Table 3.4 Accuracy on each class in HMDB-51 split 1 using different methods.

Action	DI+RGB	DF+RGB	DI+DF+RGB
brush_hair	60%	**77%**	**77%**
cartwheel	3%	**23%**	13%
catch	27%	**43%**	**43%**
chew	43%	**60%**	**60%**
clap	38%	**52%**	**52%**
climb_stairs	**60%**	53%	**60%**
climb	67%	**73%**	**73%**
dive	50%	**60%**	53%
draw_sword	30%	**47%**	40%
dribble	73%	**90%**	80%
drink	20%	**43%**	**43%**
eat	33%	**50%**	37%
fall_floor	34%	24%	**38%**
fencing	63%	**77%**	70%
flic_flac	30%	**50%**	**50%**
golf	93%	**97%**	93%
handstand	50%	**70%**	60%
hit	14%	25%	**29%**
hug	53%	**67%**	57%
jump	31%	**52%**	41%
kick_ball	37%	**40%**	33%
kick	7%	**20%**	17%
kiss	83%	83%	**87%**
laugh	50%	**70%**	57%
pick	30%	**40%**	37%
pour	83%	**97%**	93%
pullup	87%	**100%**	**100%**
punch	63%	63%	**70%**
push	70%	**87%**	77%
pushup	57%	60%	**67%**
ride_bike	93%	93%	**97%**
ride_horse	80%	**83%**	77%
run	32%	**50%**	39%
shake_hands	70%	73%	**77%**
shoot_ball	80%	**87%**	83%
shoot_bow	87%	**93%**	87%
shoot_gun	67%	60%	**70%**
sit	37%	**57%**	53%
situp	**87%**	77%	83%
smile	40%	**43%**	40%
smoke	40%	47%	**53%**
somersault	60%	**83%**	70%
stand	20%	**23%**	20%
swing_baseball	13%	**17%**	**17%**
sword_exercise	13%	**30%**	17%
sword	**31%**	17%	21%
talk	57%	**67%**	60%
throw	10%	**33%**	7%
turn	37%	**57%**	47%
walk	40%	43%	**50%**
wave	7%	**20%**	**20%**
Average	48%	**58%**	55%

Table 3.5 Accuracy comparison between AlexNet and VGG-16 on HMDB51 split 1.

Method	Alex net	VGG-16
Static RGB	40.50%[170]	47.12%
Dynamic Flow	43.69%	48.75%
Dynamic Image	40.88%	44.74%
Dynamic Flow + RGB	**53.75%**	**58.30%**
Dynamic Image + RGB	45.21%	47.96%

Table 3.6 Accuracy comparison between AlexNet and VGG-16 on UCF101 split 1.

Method	Alex net	VGG-16
Static RGB	71.20%[209]	80.00%
Dynamic Flow	75.36%	78.00%
Dynamic Flow+RGB	**84.25%**	**87.63%**
Dynamic Flow+RGB+(S)Optical flow	**88.65%**	**90.30%**

Stack(S) of optical flow + RGB (which is the standard two-stream model) on both UCF101 and HMDB51 datasets. However, interestingly, we find that, after combining stacked optical flow with DF+RGB, the accuracy improves by 3% on HMDB51 and 4% on UCF101, which shows that our new representation carries complementary information (such as long-range dynamics) that is absent previously. This shows that our three-stream model is indeed useful. Specifically, as explained earlier, our network, the static RGB data is used to capture the spatial information of the video. For temporal information, we use 10 stacked optical flow to summarize local and short-term motion features and use dynamic flow to capture long-term motion features in 25 video frames, which provide a better representation of videos and lead to improvement.

Comparisons between CNN architectures

To further validate and understand the behavior of the three-stream model, we repeated the experiment in the last section using an Alexnet architecture. As is shown in Table 3.5 and 3.6, the accuracy of VGG-16 is seen to be 2%-7% higher than for AlexNet, which is expected. We also find that the performance of DF, DF + RGB and DF + RGB + stacked optical flow show the same trend in both AlexNet and VGG-16 networks.

Benefits from Hand-crafted Features

In Table 3.7 and 3.8, we evaluate the performance of three-stream network and its combination with IDT-FV [199]. On HMDB-51, after applying IDT-FV, the accuracy of each method improves by 2% to 10%. While, using this combination also improves the accuracy for DI

Table 3.7 Accuracy comparison on HMDB51 split 1 using VGG-16 after combining with IDT-FV.

Method	+IDT-FV	Original
Static RGB	61.89%	47.06%
Dynamic Flow	58.30%	48.75%
Dynamic Image	47.96%	44.74%
Dynamic Image+RGB	65.44%	47.96%
Dynamic Flow+RGB	67.35%	58.30%
Dynamic Flow+RGB+(S)Optical flow	**67.48%**	**61.70%**
Dynamic Image+RGB+(S)Optical flow	64.13%	54.40%

Table 3.8 Accuracy comparisons on UCF101 split 1 using VGG-16 after combining with IDT-FV.

Method	+IDT-FV	Original
Dynamic Flow+RGB	**89.20%**	87.63%
Dynamic Flow+RGB+(S)Optical flow	**91.10%**	90.30%

+ RGB, this improvement is significantly inferior to DF+RGB. On the UCF-101 dataset, IDT-FV improves performance by 1– 2%. And in the process of combining with IDT-FV, we also produce the best result of our methods upon HMDB-51 and UCF-101 datasets. Because after combining IDT-FV, dynamic flow with RGB shows a promising result that is as good as the output of three-stream network (dynamic flow + stacked optical flow + RGB), we take the result of dynamic flow + RGB + IDT-FV as our best result on HMDB-51 for saving time and increasing the efficiency of networks.

Comparisons to the State of the Art

In Table 3.9, we compare our method against the state-of-the-art results in 2016. We evaluate our algorithm on both HMDB51 and UCF101 datasets. For this comparison, we use the VGG-16 model trained on dynamic flow images, combined with static RGB, a stack of 10 optical flow frames, and IDT-FV features. The results are averaged over three splits as is the standard protocol. For both the datasets, we find that our three stream model with dynamic flow images outperforms the best results using dynamic image networks [13]. For example, on HMDB51, our results by replacing the dynamic image with dynamic flow leads to a 2% improvement. We also notice a performance boost against the recent hierarchical variant [63] of the dynamic images (66.9% against 67.35%) that recursively summarizes such images from video sub-sequences.

Compared to other state-of-the-art methods, our results are very competitive. Notably, while the accuracy of two-stream fusion [62] is slightly higher than ours, our CNN archi-

Table 3.9 Classification accuracy against the state of the art (2016) on HMDB51 and UCF101 datasets averaged over three splits.

Method	HMDB51	UCF101
Two-stream [170]	59.40%	88.00%
Very Deep Two-stream Fusion [62]	**69.20%**	**93.50%**
LSTM-MSD [115]	63.57%	90.80%
IDT-FV [200]	57.20%	86.00%
IDT-HFV [144]	61.10%	87.90%
TDD+IDT-FV [209]	65.90%	91.50%
C3D + IDT-FV [188]	–	90.40%
Dynamic Image + RGB + IDT-FV [13]	65.20%	89.10%
Hierarchical Rank Pooling [66]	66.90%	91.40%
Ours		
Dyn. Flow + RGB + IDT-FV	67.19%	89.41%
Dyn. Flow+RGB+(S)Op.Flow+IDT-FV	67.35%	91.32%

tecture is significantly simpler. In Figures 3.4 and 3.5, we provide qualitative comparisons between dynamic flow and dynamic images.

3.4 Chapter Summary

In this chapter, we presented a novel video representation – dynamic flow– that summarizes a set of consecutive optical flow frames in a video sequence as a single two-channel image. We showed that our representation can compactly capture the long-range dynamics of actions. Considering this representation as an additional CNN input cue, we proposed a novel three-stream CNN architecture that incorporates single RGB frames (for action context), stack of flow images (for local action dynamics), and our novel dynamic flow stream (for long range action evolution). Experiments were provided on standard benchmark datasets (HMDB51 and UCF101) and clearly demonstrate that our method is promising in comparison to the state of the art. More importantly, our experimental results reveal that our representation captures the action dynamics more robustly than the recent dynamic image algorithm and provides complimentary information (long-range) compared to the traditional stack of flow frames. Based on our work in this chapter, another work [12] further extend this idea into an end-to-end trainable manner by introducing approximate rank pooling. In which, the fps was improved by 40 times with sacrificing 2% classification accuracy in the UCF dataset.

Fig. 3.4 Left to right: Qualitative results of RGB frames, dynamic images, and dynamic flow on UCF101 dataset.

Fig. 3.5 Left to right: Qualitative results of RGB frames, dynamic images, and dynamic flow on HMDB51 dataset.

Chapter 4

Video Representation Learning Using Discriminative Pooling

In Chapter 3, we introduce the dynamic flow image for action recognition. It applies the ordered pooling on the top of two-channel optical flow images from a video sequence to formulate an image-like video-level representation. Then, as an extra modality, we use it to train a three-stream neural network, resulting in encouraging action recognition results. However, the ranking function in this problem will treat each frame equally when doing the ordered pooling, which might be unfavourable. Because redundancy is an essential inherent feature of video data, not all frames are relevant or instructive for the end goal. If we impose equal importance for all frames, the performance would be detracted by the background and noise information. In addition, we observe that not all predictions on the short video snippets are equally informative, yet some of them must be [162]. This allows us to cast the problem in a multiple instance learning (MIL) framework, where we assume that some of the features in s given sequence are indeed useful, while the rest are not. We assume all the CNN features from a sequence (containing both the good and the bad features) to represent a positive bag, while CNN features from unrelated video frames or synthetically generated random noise frames as a negative bag. We would ideally want the features in the negative bag to be correlated well to the uninformative features in the positive bag. We then formulate a binary classification problem of separating as many good features as possible in the positive bag using a discriminative classifier (we use a support vector machine (SVM) for this purpose). The decision boundary of this classifier thus learned is then used as a descriptor for the entire video sequence, which we call the SVM Pooled (SVMP) descriptor. To accommodate the fact that we are dealing with temporally-ordered data in the positive bag, we also explore learning our representations with partial ordering relations. An illustration of our SVMP scheme is shown in the Figure 4.1.

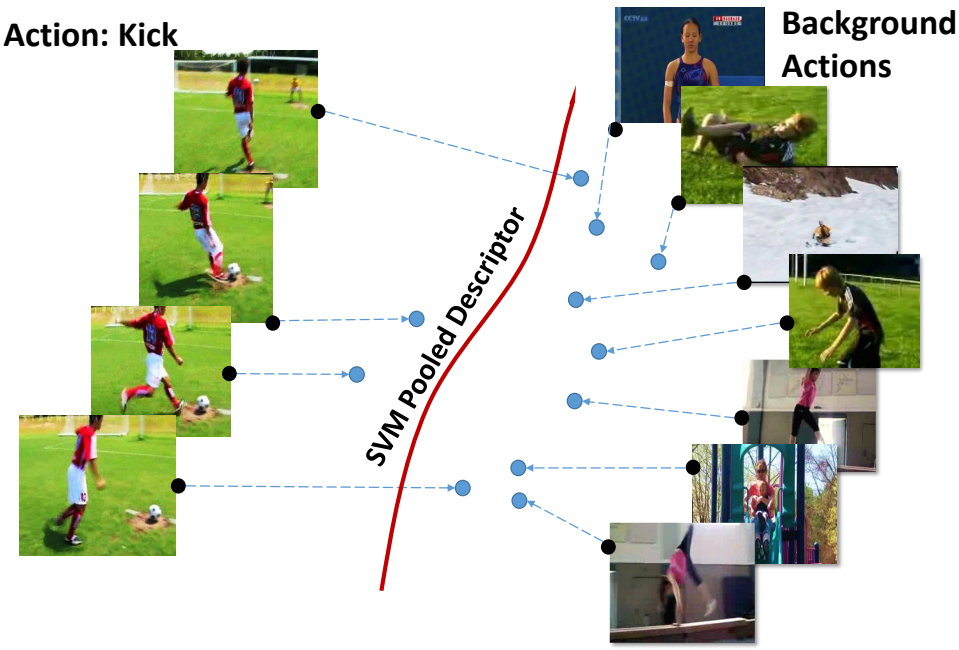


Fig. 4.1 A illustration of our discriminative pooling scheme.

Our SVMP scheme/descriptor shares several properties of standard pooled descriptors, however also showcases several important advantages. For example, similar to other pooling schemes, SVM pooling results in a compact and fixed length representation of videos of arbitrary length. Differently, our pooling gives different weights to different features. Thus, it may be seen as a type of weighted average pooling by filtering out features that are perhaps irrelevant for action recognition. Further, given that our setup uses a max-margin encoding of the features, the pooled descriptor is relatively stable with respect to data perturbations and outliers. Our scheme is agnostic to the feature extractor part of the system, for example, it could be applied to the intermediate features from any CNN model or even hand-crafted features. Moreover, the temporal dynamics of actions are explicitly encoded in the formulation. The scheme is fast to implement using publicly available SVM solvers, and also could be trained in an end-to-end manner within a CNN setup.

To evaluate our SVMP scheme, we provide extensive experiments on various datasets spanning a diverse set of tasks, namely action recognition and forecasting on HMDB-51 [107], UCF-101 [178], Kinetics-600 [101] and Charades [168]; skeleton-based action recognition on MSR action-3D [119], and NTU-RGBD [165]; image-set verification on the PubFig dataset [108], and video-texture recognition on the YUP++ dataset [61]. We outperform

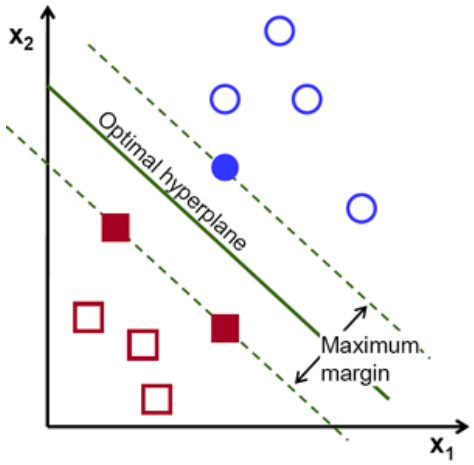


Fig. 4.2 An illustration of support vector machine, which separates two group of data points by a hyperplane.Blue circle and red square represent the data in two categories.

standard pooling methods on these datasets by a significant margin (between 3–14%) and demonstrate superior performance against others results by 1–5%.

4.1 Related work

4.1.1 Support Vector Machine

Support vector machine (SVM) is classic supervised binary machine learning models, that was first introduced by Vapnik et al. in 1963 and further developed in [44]. The support vector machine is a linear model for classification or regression problems. It can solve both linear and non-linear (by the help of kernel tricks) problems and is one of the most robust prediction methods for many practical problems. The idea of support vector machine algorithm is very straightforward, that is to find a hyperplane in the D-dimensional space that separates data points into two distinct categories. This hyperplane need to have the maximum distance with the most closed data points from two classes, where the closed data points is named as support vectors while the distance here is called margin. Thus, the support vector machine is also named as max-margin classifier. In the Figure 4.2, we demonstrate a simplified version of support vector machine. From the figure, we can see that the position of the hyperplane only depends on the support vectors from entire data points. As a result, we only include the support vectors during the calculation and this property is called sparsity, which make the support vector machine algorithm more efficiency, especially when it applies non-linear kernel tricks.

Formally, given the data points $X = (x_1, x_2, ..., x_n)$, where $x_i \in \mathbb{R}^D$ is the i^{th} point in D dimensional space. The SVM objective can be written as:

$$\begin{aligned} &min\ \frac{1}{2}\|w\|^2 \\ &\text{subject to}:\ 1 - y^i(w^T x_i - b) \leq 0,\ \forall x_i \in X \end{aligned} \tag{4.1}$$

where w and b is the decision boundary and bias of the SVM, $y^i \in \{-1, +1\}$ is the class label for binary classes. In this case, it is assumed that the data point is linearly separable. To extend the SVM to cases in which the data are not linearly separable, the hinge loss function is introduced. The Equation 4.1 can be rewritten as:

$$min\ \frac{\lambda}{2}\|w\|^2 + \frac{1}{n}\sum_i max(0, 1 - y^i(w^T x_i - b)) \tag{4.2}$$

where the λ determine the trade-off between the margin size and classification accuracy.

When people investigate the duality of SVM optimization, the above formulation is called its prime problem. Specifically, we can solve the Equation 4.2 by introducing Lagrange multiplier, because it is quadratic programming. Then we will have:

$$\mathcal{L}(w, b, \alpha) = \frac{\lambda}{2}\|w\|^2 - \sum_i \alpha_i [y^i(w^T x + b) - 1] \tag{4.3}$$

where the α is the Lagrange multiplier.

To solve the above equation following the generalized Lagrange (consider its Lagrange duality), its solution has to satisfy Karush-Kuhn-Tucker (KKT) conditions. Then, we will have:

$$\frac{\partial \mathcal{L}}{\partial w} = 0\ \rightarrow\ w = \sum_i \alpha_i y^i x_i \tag{4.4}$$

$$\frac{\partial \mathcal{L}}{\partial b} = 0\ \rightarrow\ 0 = \sum_i \alpha_i y^i \tag{4.5}$$

Plugging these two equations into the Equation 4.3 and reduce it to its dual problem, we can get:

$$\begin{aligned} &\mathcal{L}(w, b, \alpha) = \sum_i \alpha_i + \frac{1}{2}\sum_{ij} \alpha_i \alpha_j y^i y^j x_i^T x_j \\ &\sum_i \alpha_i y^i = 0,\ \alpha_i \geq 0 \end{aligned} \tag{4.6}$$

Since $w = \sum_i \alpha_i y^i x_i$, we will have:

$$w^T x + b = \sum_i \alpha_i y^i x_i^T x + b \tag{4.7}$$

From the above equation, it is found that the decision boundary only depends on the inner product of data points. As a result, we could map the inner product $< x_i, x >$ to another space by using the kernel embedding, which enable the linearly non-separable data points linearly separable. Moreover, due to the sparsity nature of the support vector machine, the kernel-based calculation is very efficient.

Intuition of Using Decision Boundary as Video Representation

When we apply support vector machine into video representation learning, each x_t is the frame-level feature point at time step t. As previously stated, one of the characteristics of video data is its redundancy. It's difficult to learn discriminative characteristics from the video frames since there's so much repetition. In addition, the deep CNN features is usually high dimensional (from hundreds to thousands) and non-interpretable. We believe there are two key criteria for learning more discriminative video representation based on these two facts: a) not all frames from the video sequence should be equally included. b) not all dimension from the deep CNN feature vector should be equally treated.

Recall equations of support vector machine, it is found that some features of the SVM decision boundary are able to help us to get more discriminative video representation. From the Equation 4.2, the decision boundary of support vector machine is essentially a weighting vector associated with each dimension of the support feature vectors, which equivalent to weighting the data points according to the distribution of data points in the SVM classification. Moreover, from the dual problem of support vector machine, we have $w = \sum_i \alpha_i y^i x_i$, in which the decision boundary is the weighted average of support vectors. These two properties of the SVM decision boundary meet the requirements we described above. However, the support vectors, as well as the negative class of the training data, have a significant impact on the distribution of the decision border. Using the decision boundary as the video representation will help us achieve better discriminative video representations when the negative data points are intelligently picked. The discussion of how to pick the negative class training data will be introduced in this Chapter and Chapter 5.

4.1.2 Multiple Instance Learning

An important component in this Chapter is the MIL scheme, which is a popular data selection technique [42, 117, 216, 224, 229]. In the context of video representation, schemes similar in motivation have been suggested before. For example, Satkin and Hebert [161] explore

the effect of temporal cropping of videos to regions of actions; however, it assumes these regions are continuous. Nowozin et al. [138] represent videos as sequences of discretized spatiotemporal sets and reduces the recognition task into a max-gain sequence finding problem on these sets using an LPBoost classifier. Similar to ours, Li et al. [118] propose an MIL setup for complex activity recognition using a dynamic pooling operator – a binary vector that selects input frames to be part of an action, which is learned by reducing the MIL problem to a set of linear programs. Chen et al. [181] and Wang et al. [210] propose a latent variable based model to explicitly localize discriminative video segments where events take place. Vahdat et al. present a compositional model in [194] for video event detection, which is presented using a multiple kernel learning based latent SVM. While all these schemes share similar motivations as ours, we cast our MIL problem in the setting of normalized set kernels [69] and reduce the formulation to standard SVM setup which can be solved rapidly. In the $\propto$-SVMs of Yu et al., [109, 225], the positive bags are assumed to have a fixed fraction of positives, which is a criterion we also assume in our framework. However, the negative bag selection, optimization setup and our goals are different; specifically, our goal is to learn a video representation for any subsequent task including recognition, anticipation, and detection, while the framework in [109] is designed for event detection. And we generate the negative bag by using CNN features generated via inputting random noise images to the network.

4.2 Approach

In this section, we first describe the problem of learning SVMP descriptors and introduce three different ways to solve it. Before proceeding, we provide a snapshot of our main idea and problem setup graphically in Figure 4.3. Starting from frames (or flow images) in positive and negative bags, these frames are first passed through some CNN model for feature generation. These features are then passed to our SVMP module that learns (non-linear) hyperplanes separating the features from the positive bag against the ones from the negative bag, the latter is assumed fixed for all videos. These hyperplane representations are then used to train an action classifier at the video level. In the following, we formalize these ideas concretely.

4.2.1 Problem Setup

Let us assume we are given a dataset of N video sequences $\mathscr{X}^+ = \left\{X_1^+, X_2^+, ..., X_N^+\right\}$, where each X_i^+ is a set of frame level features, *i.e.*, $X_i^+ = \left\{\mathbf{x}_1^{i+}, \mathbf{x}_2^{i+}, ..., \mathbf{x}_n^{i+}\right\}$, each $\mathbf{x}_k^{i+} \in \mathbb{R}^p$. We

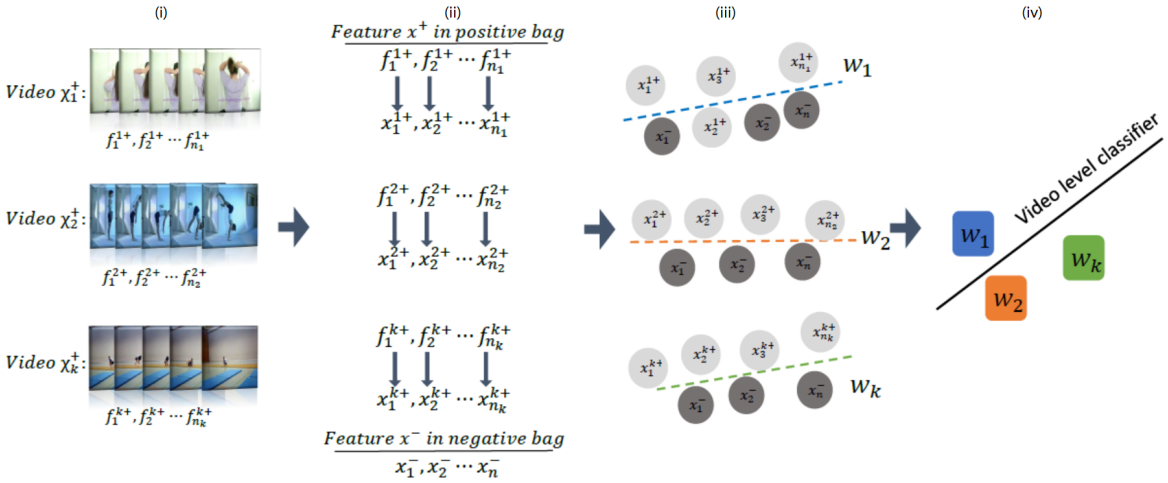


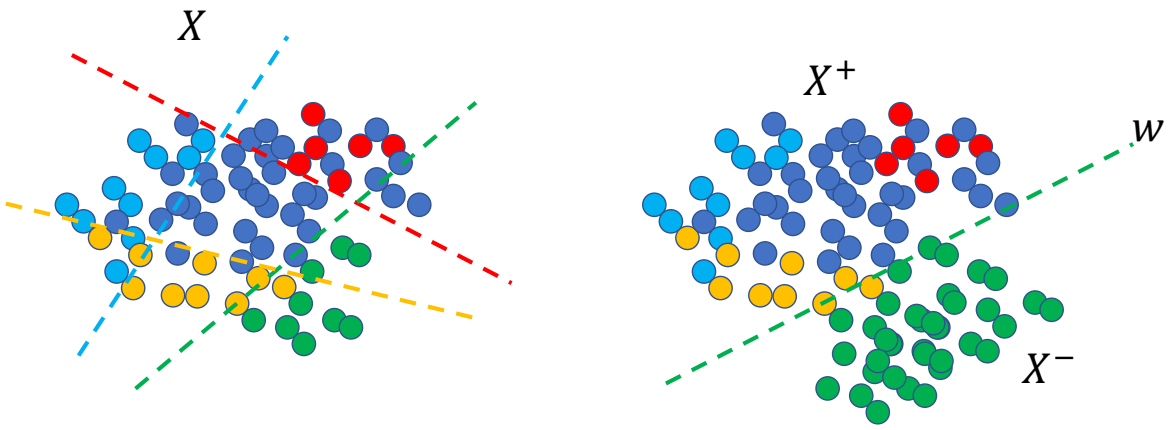
Fig. 4.3 Illustration of SVM Pooling pipeline. (i) Extraction of frames from videos, (ii) Converting frames f into feature x, (iii) Learning decision boundary w from feature x, and (iv) Using w as video descriptor.

assume that each X_i^+ is associated with an action class label $y_i^+ \in \{1,2,...,d\}$. Further, the $+$ sign denotes that the features and the sequences represent a positive bag. We also assume that we have access to a set of sequences $\mathscr{X}^- = \{X_1^-, X_2^-, ...X_M^-\}$ belonging to actions different from those in $\mathscr{X}^+$, where each $X_j^- = \left\{\mathbf{x}_1^{j-}, \mathbf{x}_2^{j-}, ..., \mathbf{x}_n^{j-}\right\}$ are the features associated with a negative bag, each $\mathbf{x}_k^{j-} \in \mathbb{R}^p$. For simplicity, we assume all sequences have same n number of features. Further note that our scheme is agnostic to the type of features, i.e., the feature may be from a CNN or are hand-crafted.

Our goals are two-fold, namely (i) to learn a classifier decision boundary for every sequence in $\mathscr{X}^+$ that separates a fraction η of them from the features in $\mathscr{X}^-$ and (ii) to learn video level classifiers on the classes in the positive bags that are represented by the learned decision boundaries in (i). In the following, we will provide a multiple instance learning formulation for achieving (i), and a joint objective combining (i) and learning (ii). However, before presenting our scheme, we believe it may be useful to gain some insights into the main motivations for our scheme.

As alluded to above, given the positive and negative bags, our goal is to learn a linear (or non-linear) classification boundary that could separate the two bags with a classification accuracy of $\eta\%$ – this classification boundary is used as the descriptor for the positive bag. Referring to the conceptual illustration in Figure 4.4(a), when no negative bag is present, there are several ways to find a decision hyperplane in a max-margin setup that could potentially satisfy the η constraint. However, there is no guarantee that these hyperplanes are useful for action recognition. Instead, by introducing a negative bag, which is almost

certainly to contain irrelevant features, it may be easier for the decision boundary to identify useless features from the rest; the latter containing useful action related features, as shown in Figure 4.4(b). This is precisely our intuitions for proposing this scheme.



(a) (b)

Fig. 4.4 An illustration of our overall idea. (a) the input data points, and the plausible hyperplanes satisfying some η constraint, (b) when noise $\mathscr{X}^-$ is introduced (green dots), it helps identify noisy features/data dimensions, towards producing a hyperplane w that classifies useful data from noise, while satisfying the η constraint.

4.2.2 Learning Decision Boundaries

As described above, our goal in this section is to generate a descriptor for each sequence $X^+ \in \mathscr{X}^+$; this descriptor we define to be the learned parameters of a hyperplane that separates the features $\mathbf{x}^+ \in X^+$ from all features in $\mathscr{X}^-$. We do not want to warrant that all $\mathbf{x}^+$ can be separated from $\mathscr{X}^-$ (since several of them may belong to a background class), however we assume that at least a fixed fraction η of them are classifiable. Mathematically, suppose the tuple (w_i, b_i) represents the parameters of a max-margin hyperplane separating some of the features in a positive bag X_i^+ from all features in $\mathscr{X}^-$, then we cast the following objective, which is a variant of the sparse MIL (SMIL) [18], normalized set kernel (NSK) [69],

and $\propto$-SVM [225] formulations:

$$\underset{w_i\in\mathbb{R}^p, b_i\in\mathbb{R}, \zeta\geq 0}{\arg\min} \; P1(w_i,b_i) := \frac{1}{2}\|w_i\|^2 + C_1 \sum_{k=1}^{(M+1)n} \zeta_k \tag{4.8}$$

$$\text{subject to } \theta(\mathbf{x};\eta)\left(w_i^T\mathbf{x}+b_i\right) \geq 1-\zeta_k \tag{4.9}$$

$$\theta(\mathbf{x};\eta) = -1, \forall \mathbf{x} \in \left\{X_i^+ \bigcup \mathscr{X}^-\right\} \setminus \hat{X}_i^+ \tag{4.10}$$

$$\theta(\hat{\mathbf{x}};\eta) = 1, \forall \hat{\mathbf{x}} \in \hat{X}_i^+ \tag{4.11}$$

$$\left|\hat{X}_i^+\right| \geq \eta \left|X_i^+\right|. \tag{4.12}$$

In the above formulation, we assume that there is a subset $\hat{X}_i^+ \subset X_i^+$ that is classifiable, while the rest of the positive bag need not be, as captured by the ratio in (4.12). The variables ζ capture the non-negative slacks weighted by a regularization parameter C_1, and the function θ provides the label of the respective features. Unlike SMIL or NSK objectives, that assumes the individual features $\mathbf{x}$ are summable, our problem is non-convex due to the unknown set $\hat{X}^+$. However, this is not a serious deterrent to the usefulness of our formulation and can be tackled as described in the sequel and supported by our experimental results.

As our formulation is built on an SVM objective, we call this specific discriminative pooling scheme as *SVM pooling* and formally define the descriptor for a sequence as:

Definition 1 (SVM Pooling Desc.)**.** *Given a sequence X of features $\mathbf{x}\in\mathbb{R}^p$ and a negative dataset $\mathscr{X}^-$, we define the* SVM Pooling *(SVMP) descriptor as* $\text{SVMP}(X) = [w,b]^T \in \mathbb{R}^{p+1}$*, where the tuple (w,b) is obtained as the solution of problem $P1$ defined in* (4.8)*.*

4.2.3 Optimization Solutions

The problem $P1$ could be posed as a mixed-integer quadratic program (MIQP), which is unfortunately known to be in nondeterministic polynomial time (NP) [111]. The problem $P1$ is also non-convex due to the proportionality constraint η, and given that the labels $\theta(\mathbf{x};\eta)$ are unknown. Towards a practically useful approximate solution circumventing these difficulties, we present three optimization strategies below.

Exhaustive Enumeration

A naïve way to solve problem $P1$ could be to enumerate all the possible $\theta(\mathbf{x};\eta)$ that meet a given η constraint, which reduces solving the problem $P1$ to the classical SVM problem for each instantiation of the plausible θ assignments. In such a setting, for a given sequence, we

can rewrite (4.8) as:

$$\underset{w_i\in\mathbb{R}^p, b_i\in\mathbb{R}, \zeta\geq 0}{\arg\min} \frac{1}{2}\|w_i\|^2 + C_1 \sum_{k=1}^{(M+1)n} \zeta_k + \max(0, 1-\zeta_k - \theta(\mathbf{x};\eta)(w_i^T\mathbf{x}+b_i)), \tag{4.13}$$

where the constraints are included via the hinge loss. Once these subproblems are solved, we could compare the optimal solutions for the various subsets of the positive bag, and pick the best solution with smallest objective value. As is apparent, this naïve strategy becomes problematic for longer sequences or when η is not suitably chosen.

Alternating algorithm

This is a variant of the scheme proposed in [225]. Instead of enumerating all possible $\theta(\mathbf{x};\eta)$ as above, the main idea here is to fix $\theta(\mathbf{x};\eta)$ or $[w,b]$ alternately and optimize the other. The detailed algorithm is shown in the Alg. 1.

Input: X^+, $\mathscr{X}^-$, η
Initialize θ *according to* η;
repeat
  Fix θ *to solve* $[w,b] \leftarrow \text{SVM}(X^+, \mathscr{X}^-, \theta)$;
  Fix $[w,b]$ *to solve* θ: *Reinitialize* $\theta_i \leftarrow -1, \forall i \in (i,n)$;
  for $i = 1 \rightarrow n$ **do**
    Set $\theta_i \leftarrow 1$;
    record the reduction of Objective
  end
  Sort and select the top R reductions, $R = \eta n$;
  Get θ *according to the sorting*;
until *Reduction is smaller than a threshold* (10^{-4});
return $[w,b]$

Algorithm 1: Alternating solution to the MIL problem $P1$

In the Algorithm 1, fixing θ to solve $[w,b]$ is a standard SVMP problem as in the enumeration algorithm above. When fixing $[w,b]$ to solve θ, we apply a similar strategy as in [225]; i.e., to initialize all labels in θ as -1, and then to turn each θ_i to $+1$ and record the reduction in the objective. Next, we sort these reductions to get the top R best reductions, where $R = \eta n$. A higher reduction implies it may lead to a smaller objective. Next, these top R θ_i will be set to $+1$ in θ. While, there is no theoretical guarantee for this scheme to converge to a fixed point, empirically we observe a useful convergence, which we limit via a suitable threshold.

Parameter-tuning algorithm

As is clear, both the above schemes may be computationally expensive in general. We note that the regularization parameter C_1 in (4.8) controls the positiveness of the slack variables ζ, thereby influencing the training error rate. A smaller value of C_1 allows more data points to be misclassified. If we make the assumption that useful features from the sequences are easily classifiable compared to background features, then a smaller value of C_1 could help find the decision hyperplane easily (further assuming the negative bag is suitably chosen). However, the correct value of C_1 depends on each sequence. Thus, in Algorithm (2), we propose a heuristic scheme to find the SVMP descriptor for a given sequence X^+ by iteratively tuning C_1 such that at least a fraction η of the features in the positive bag are classified as positive.

Input: X^+, $\mathscr{X}^-$, η
$C_1 \leftarrow \varepsilon$, $\lambda > 1$;
repeat
  $C_1 \leftarrow \lambda C_1$;
  $[w,b] \leftarrow \arg\min_{w,b} \mathrm{SVM}(X^+, \mathscr{X}^-, C_1)$;
  $\hat{X}^+ \leftarrow \{\mathbf{x} \in X^+ \mid w^T\mathbf{x} + b \geq 0\}$;
until $\frac{|\hat{X}^+|}{|X^+|} \geq \eta$;
return $[w,b]$

Algorithm 2: Parameter-tuning solution for MIL problem $P1$

A natural question here is how optimal is this heuristic? Note that, each step of Algorithm (2) solves a standard SVM objective. Suppose we have an oracle that could give us a fixed value C for C_1 that works for all action sequences for a fixed η. As is clear, there could be multiple combinations of data points in $\hat{X}^+$ that could satisfy this η (as we explored in the Enumeration algorithm above). If $\hat{X}_p^+$ is one such $\hat{X}^+$. Then, $P1$ using $\hat{X}_p^+$ is just the SVM formulation and is thus convex. Different from previous algorithms, in Alg. 2, we adjust the SVM classification rate to η, which is easier to implement. Assuming we find a C_1 that satisfies the η-constraint using $P1$, then due to the convexity of SVM, it can be shown that the optimizing objective of P1 will be the same in both cases (exhaustive enumeration and our proposed regularization adjustment), albeit the solution $\hat{X}_p^+$ might differ (there could be multiple solutions).

4.2.4 Nonlinear Extensions

In problem $P1$, we assume a linear decision boundary generating SVMP descriptors. However, looking back at our solutions in Algorithms (1) and (2), it is clear that we are dealing with

standard SVM formulations to solve our relaxed objectives. In the light of this, instead of using linear hyperplanes for classification, we may use nonlinear decision boundaries by using the kernel trick to embed the data in a Hilbert space for better representation. Assuming $\mathscr{X} = \mathscr{X}^+ \cup \mathscr{X}^-$, by the Representer theorem [174], it is well-known that for a kernel $K : \mathscr{X} \times \mathscr{X} \to \mathbb{R}_+$, the decision function f for the SVM problem P1 will be of the form:

$$f(.) = \sum_{\mathbf{x} \in X^+ \cup \mathscr{X}^-} \alpha_{\mathbf{x}} K(., \mathbf{x}), \tag{4.14}$$

where $\alpha_{\mathbf{x}}$ are the parameters of the non-linear decision boundaries. However, from an implementation perspective, such a direct kernelization may be problematic, as we will need to store the training set to construct the kernel. We avoid this issue by restricting our formulation to use only homogeneous kernels [196], as such kernels have explicit linear feature map embeddings on which a linear SVM can be trained directly. This leads to exactly the same formulations as in (4.8), except that now our features $\mathbf{x}$ are obtained via a homogeneous kernel map. In the sequel, we call such a descriptor a *nonlinear SVM pooling* (NSVMP) descriptor.

4.2.5 Temporally-Ordered Extensions

In the formulations we proposed above, there are no explicit constraints to enforce the temporal order of features in the SVMP descriptor. This is because, in the above formulations, we assume the features themselves capture the temporal order already. For example, the temporal stream in a two-stream model is already trained on a densely-sampled stack of consecutive optical flow frames. However, motivated by several recent works [13, 31, 64, 204], we extend our Equation (4.8) by including ordering constraints as:

$$w^T \mathbf{x}_j^{i+} + \delta \leq w^T \mathbf{x}_k^{i+}, \quad \forall j < k; \mathbf{x}_j^{i+}, \mathbf{x}_k^{i+} \in \hat{X}_i^+ \tag{4.15}$$

where we reuse the notation defined above and define $\delta > 0$ as a margin enforcing the order. In the sequel, we use this temporally-ordered variant of SVMP for our video representation. Note that with the ordering constraints enforced, it is difficult to use the enumerative or alternating schemes for finding the SVMP descriptors, instead we use Alg. 2 by replacing the SVM solver by a custom solver [16].

4.3 End-to-End CNN Learning

In this section, we address the problem of training a CNN end-to-end with SVM pooling as an intermediate layer – the main challenge is to derive the gradients of SVMP for efficient backpropagation. This challenge is amplified by the fact that we use the parameters of the decision hyperplane to generate our pooling descriptor, this hyperplane is obtained via a non-differentiable argmin function (refer to (4.8)). However, fortunately, there is well-developed theory addressing such cases using the implicit function theorem [51], and several recent works towards this end in the CNN setting [74]. We follow these approaches and derive the gradients of SVMP below.

4.3.1 Discriminative Pooling Layer

In Figure 4.5, we describe two ways to insert the discriminative pooling layer into the CNN pipeline, namely (i) inserting SVMP at some intermediate layer and (ii) inserting SVMP at the end of the network just before the final classifier layer. While the latter pools smaller dimensional features, computing the gradients will be faster (as will be clear shortly). However, the last layer might only have discriminative action features alone, and might miss other spatio-temporal features that could be useful for discriminative pooling. This is inline with our observations in our experiments in Section 4.4 that suggest that applying discriminative pooling after pool5 or fc6 layers is significantly more useful than at the end of the fc8 layer. This choice of inserting the pooling layer between some intermediate layers of the CNN leads to the first choice. Figure 4.5 also provides the gradients that need to be computed for back-propagation in either case. The only new component of this gradient is that for the argmin problem of pooling, which we derive below.

4.4 Experiments

In this section, we explore the utility of discriminative pooling on several vision tasks, namely (i) action recognition using video and skeletal features, (ii) localizing actions in videos, (iii) image set verification, and (iv) recognizing dynamic texture videos. We introduce the respective datasets and experimental protocols in the next.

4.4.1 Data Preprocessing

HMDB-51 [107] and UCF-101 [178]: For these datasets, we analyze different combinations of features on multiple CNN frameworks.

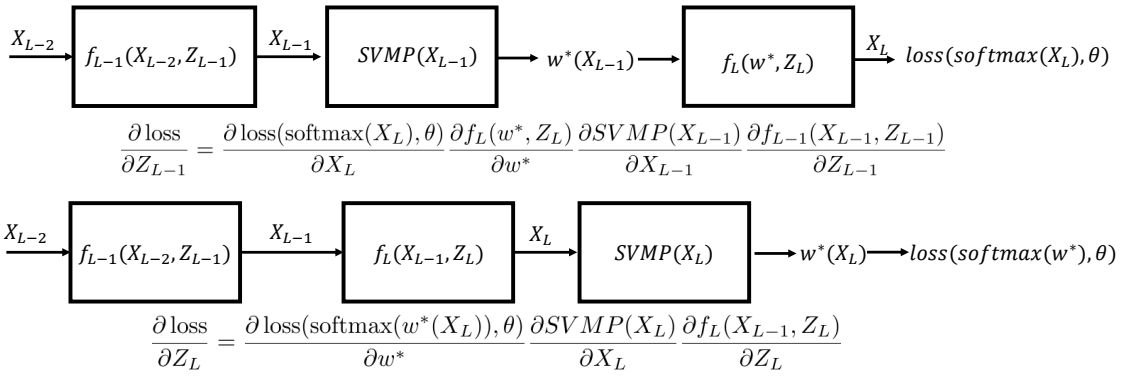


Fig. 4.5 Two possible ways to insert SVM pooling layer within a standard CNN architecture. In the first option (top), we insert the SVMP layer between fully connected layers, while in the latter we include it before the final classifier layer. The choice of $L-1$ layer for the former is arbitrary. We also show the corresponding partial gradients with respect to weights of the layer penultimate to the SVM pooling layer. Except for the gradients $\frac{\partial SVMP(X)}{\partial X}$, other gradients are the standard ones. Here Z_ℓ represents the weights of the ℓ-th layer of the network.

Charades [168]: Using the provided two-stream fc7 feature[1], we evaluate the performance on both tasks using mean average precision (mAP) on the validation set.

Kinetics-600 [101]: We apply our SVMP scheme on the CNN features (2048-D) extracted from the I3D network [24].

MPII Cooking Activities Dataset [154]: To analyze how SVMP works with hand-crafted features, we show experiments using the publicly available 4000-D bag-of-words features computed over HOG and trajectory features from this dataset and report mean average precision (mAP) after 7-fold cross-validation.

MSR Action3D [119] and NTU-RGBD [165]: To analyze the performance of SVMP on non-linear features, we use a lie-algebra encoding of the skeletal data as proposed in [197] for the MSR dataset. As for NTU-RGBD, we use a temporal CNN as in [104], but uses SVMP instead of their global average pooling.

Public Figures Face Database (PubFig) [108]: To generate features, we fine-tune a ResFace-101 network [128] on this dataset and follow the evaluation protocol of [82].

YUP++ dataset [61]: In this dataset, we use the latest Inception-ResNet-v2 model [183] to generate features (from last dense layer) from RGB frames and evaluate the performance according to the setting in [61], which use a 10/90 train-test ratio.

[1] http://vuchallenge.org/charades.html

4.4.2 Parameter Analysis

In this section, we analyze the influence of each of the parameters in our scheme.

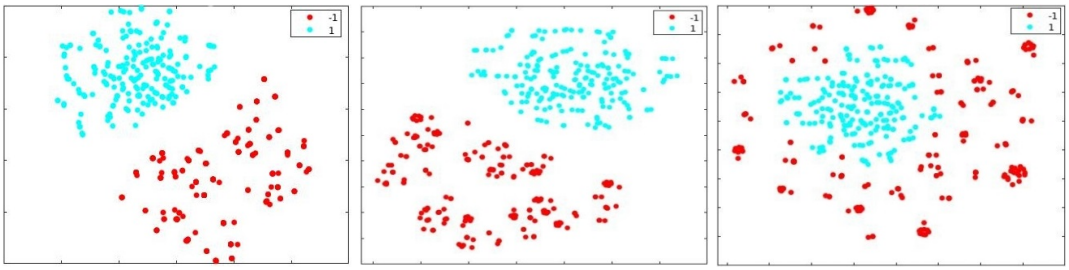
Selecting Negative Bags: An important step in our algorithm is the selection of the positive and negative bags in the MIL problem. We randomly sample the required number of frames (say, 50) from each sequence/fold in the training/testing set to define the positive bags. In terms of the negative bags, we need to select samples that are unrelated to the ones in the positive bags. We explored four different negatives in this regard to understand the impact of this selection. We compare our experiments on the HMDB-51 (and UCF101) datasets. Our considered the following choices for the negative bgs: clips from (ithe ActivityNet dataset [19] unrelated to HMDB-51, (ii) the UCF-101 dataset unrelated to HMDB-51, (iii) the Thumos Challenge background sequences[2], and (iv) synthesized random white noise image sequences. For (i) and (ii), we use 50 frames each from randomly selected videos, one from every unrelated class, and for (iv) we used 50 synthesized white noise images, and randomly generated stack of optical flow images.

Specifically, for the latter, we pass white noise RGB images to the same CNN models and extract the feature from the last fully-connected layer. As for hand-crafted or geometry features used in our other experiments (such as action recognition on human pose sequences), we directly use the white noise as the negative bag. As shown in Figure 4.8(a), the white noise negative is seen to showcase better performance for both lower and higher value of η parameter.

To understand this trend, in Figure 4.6, we show TSNE plots visualizing the deep CNN features for the negative bag variants. Given that the CNNs are trained on real-world image data and we extract features from the layer before the last linear layer, it is expected that these features be linearly separable (as seen in Figure 4.6(a) and 4.6(b)). However, we believe using random noise inputs may be activating combinations of filters in the CNN that are never co-activated during training, resulting in features that are highly non-linear (as Figure 4.6(c) shows). Thus, when requiring SVMP to learn linear/non-linear decision boundaries to classify video features against these "noise" features perhaps forces the optimizer to select those dimensions in the inputs (positive bag) that are more correlated with actions in the videos, thereby empowering the descriptor to be more useful for classification.

In Figure 4.7, we show the TSNE visualizations of SVMP descriptors comparing to average pooling and max pooling on data from 10-classes of HDMB-51 dataset. The visualization shows that SVMP leads to better separated clusters, substantiating that SVMP is learning discriminative representations.

[2]http://www.thumos.info/home.html



(a) Thumos (b) UCF101 (c) White Noise

Fig. 4.6 T-SNE plots of positive (blue) and negative bags (red) when using negatives from: (a) Thumos, (b) UCF101, and (c) white noise.

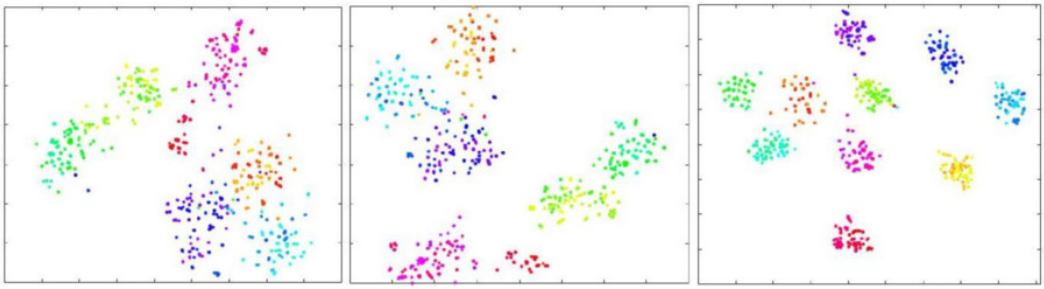


Fig. 4.7 T-SNE visualizations of SVMP and other pooling methods on sequences from the HMDB51 dataset (10 classes used). From left to right, Average Pooling, Max Pooling, and SVMP.

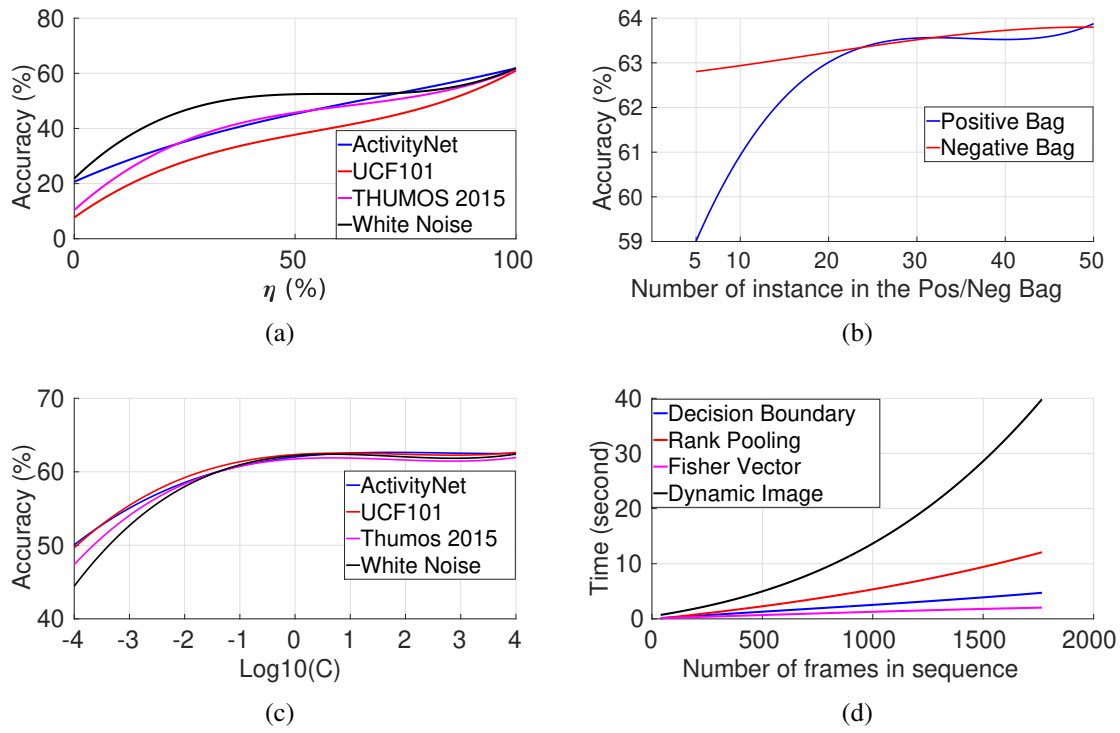


Fig. 4.8 Analysis of the parameters used in our scheme. All experiments use VGG features from fc6 dense layer. See text for details.

Choosing Hyperparameters: The three important parameters in our scheme are (i) the η deciding the quality of an SVMP descriptor, (ii) $C_1 = C$ used in Algorithm 2 when finding SVMP per sequence, and (iii) sizes of the positive and negative bags. To study (i) and (ii), we plot in Figures 4.8(c) and 4.8(a) for HMDB-51 dataset, classification accuracy when C is increased from 10^{-4} to 10^4 in steps and when η is increased from 0-100% and respectively. We repeat this experiment for all the different choices of negative bags. As is clear, increasing these parameters reduces the training error, but may lead to overfitting. However, Figure 4.8(b) shows that increasing C increases the accuracy of the SVMP descriptor, implying that the CNN features are already equipped with discriminative properties for action recognition. However, beyond $C = 10$, a gradual decrease in performance is witnessed, suggesting overfitting to bad features in the positive bag. Thus, we use $C = 10$ (and $\eta = 0.9$) in the experiments to follow. To decide the bag sizes for MIL, we plot in Figure 4.8(b), performance against increasing size of the positive bag, while keeping the negative bag size at 50 and vice versa; i.e., for the red line in Figure 4.8(b), we fix the number of instances in the positive bag at 50; we see that the accuracy raises with the cardinality of the negative bag. A similar trend, albeit less prominent is seen when we repeat the experiment with the negative bag size, suggesting that about 30 frames per bag is sufficient to get a useful descriptor.
Running Time: In Figure 4.8(d), we compare the time it took on average to generate SVMP descriptors for an increasing number of frames in a sequence on the UCF101 dataset. For comparison, we plot the running times for some of the recent pooling schemes such as rank pooling [13, 64] and the Fisher vectors [200]. The plot shows that while our scheme is slightly more expensive than standard Fisher vectors (using the VLFeat[3]), it is significantly cheaper to generate SVMP descriptors than some of the recent popular pooling methods. To be comparable, we use publicly available code of SVM in SVMP as well as in rank pooling.

4.4.3 Experiments on HMDB-51 and UCF-101

Following recent trends, we use a two-stream CNN model in two popular architectures, the VGG-16 and the ResNet-152 [62, 171]. For the UCF101 dataset, we directly use publicly available models from [62]. For the HMDB dataset, we fine-tune a two-stream VGG/ResNet model trained for the UCF101 dataset. Note that, even though we use a VGG/ResNet model in our framework, our scheme is general and could use any other features or CNN architectures, which will be demonstrated in the experiments of other datasets.
Model Training: We follow the standard pre-processing pipeline as described in [60]. For the HMDB dataset, both spatial and temporal networks are fine-tuned from UCF101 models

[3]http://www.vlfeat.org/

with an initial learning rate of 10^{-4}. The network is trained using SGD with a momentum of 0.9 and weight decay of 0.0005. We use mini-batches of 64 frames. For computing the decision boundaries, we use the features from the intermediate CNN layers as explained in the sequel.and high drop-out in VGG-16 model to prevent over-fitting

In the two-stream model, the spatial stream uses single RGB frames as input. To this end, we first resize the video frames to make the smaller side equal to 256. Further, we augment all frames via random horizontal flips and random crops to a 224 x 224 region. For temporal stream, which takes a stack of optical flow images as input, we choose the OpenCV implementation of the TV-L1 algorithm for flow computation [227]. This follows generating a 10-channel stack of flow images as input to the CNN models. For every flow image stack, we subtract the median value, to reduce the impact of camera motion, followed by thresholding them in the range of ± 20 pixels and setting every other flow vector outside this range to zero, thereby removing outliers.

As alluded to above, for the HMDB dataset, both spatial and temporal networks are fine-tuned from the respective UCF101 models with an initial learning rate of 10^{-4} and high drop-out in VGG-16 model to prevent over-fitting. The network is trained using SGD with a momentum of 0.9 and weight decay of 0.0005. We use mini-batches of size 64 for training. For computing the decision boundaries, we use the features from the intermediate CNN layers as explained in the sequel.

SVMP Optimization Schemes: We proposed three different optimization strategies for solving our formulation (Section 4.2.3). The enumerative solution is trivial and non-practical. Thus, we will only compare Algorithms 1 and 2 in terms of the performance and efficiency. In Table 4.1, we show the result between the two on fc6 features from a VGG-16 model. It is clear that the alternating solution is slightly better than parameter-tuning solution; however, is also more computationally expensive. Considering the efficiency, especially for the large-scale datasets, we use parameter-tuning solution in the following experiments.

SVMP on Different CNN Features: We generate SVMP descriptors from different intermediate layers of the CNN models and compare their performance. Specifically, features from each layer are used as the positive bags and SVMP descriptors computed using Alg. 1 against the chosen set of negative bags. In Table 4.2, we report results on split-1 of the HMDB dataset and find that the combination of fc6 and pool5 gives the best performance for the VGG-16 model, while pool5 features alone show good performance using ResNet. We thus use these feature combinations for experiments to follow.

Linear vs Non-Linear SVMP: We analyze the complementary nature of SVMP and its non-linear extension NSVMP (using a homogeneous kernel) on HMDB-51 and UCF-101 split1. The results are provided in Table 4.3, we concatenate the feature from selected layer

Table 4.1 Comparison between Algorithms 1 and 2 in HMDB-51 split-1.

Method	Accuracy	Avg. Time (sec)/Video
Alternating Algorithm (Alg. 1)	**69.8%**	2.4
Parameter-tuning Algorithm (Alg. 2)	69.5%	**0.2**

Table 4.2 Comparison of SVMP descriptors using various CNN Features on HMDB-51 split-1.

Feature/ model	Accuracy independently	Accuracy when combined with:
pool5 (vgg-16)	57.9%	**63.8%** (fc6)
fc6 (vgg-16)	63.3%	-
fc7 (vgg-16)	56.1%	57.1% (fc6)
fc8 (vgg-16)	52.4%	58.6% (fc6)
softmax (vgg-16)	41.0%	46.2% (fc6)
pool5 (ResNet-152)	**69.5%**	-
fc1000 (ResNet-152)	61.1%	68.8% (pool5)

in spatial and temporal stream, and apply both linear and non-linear SVMP. It is obvious that the non-linear SVMP descriptor is complementary to the linear one. Thus, we will use the combination, namely SVMP, for the following experiments.

End-to-End Learning and Ordered-SVMP: In Table 4.4[4], we compare to the end-to-end learning setting as described in Section 4.3. For end-to-end learning, we insert our discriminative pooling layer after the 'fc6' layer in VGG-16 model and the 'pool5' layer in ResNet model. We also present results when using the temporal ordering constraint (TC) into the SVMP formulation to build the ordered-SVMP. From the results, it appears that although the soft-attention scheme performs better than average pooling, it is inferior to SVMP itself; which is unsurprising given it does not use a max-margin optimization. Further, our end-to-end SVMP layer is able to achieve similar (but slightly inferior) performance to SVMP, which perhaps is due to the need to approximate the Hessian. As the table shows,

[4]All experiments in Table 4.4 use the same input features.

Table 4.3 Comparison between SVMP and NSVMP on split-1.

	HMDB-51		UCF-101	
	VGG	ResNet	VGG	ResNet
linear-SVMP	63.8%	69.5%	91.6%	92.2%
nonlinear-SVMP	64.4%	69.8%	92.0%	93.1%
Combination	**66.1%**	**71.0%**	**92.2%**	**94.0%**

we found that the temporal ranking is indeed useful for improving the performance of naïve SVMP. Thus, in the following experiments, we use SVMP with temporal ranking for all video-based tasks.

Table 4.4 Comparison to standard pooling methods on split-1. TC is short for Temporal Constraint, E2E is short for end-to-end learning.

	HMDB-51		UCF-101	
	VGG	ResNet	VGG	ResNet
Spatial Stream-AP[59, 62]	47.1%	46.7%	82.6%	83.4%
Spatial Stream-MP	46.5%	45.1%	82.2%	82.7%
Spatial Stream-SVMP	58.3%	57.4%	85.7%	87.6%
Spatial Stream-SVMP(E2E)	56.4%	55.1%	83.2%	85.7%
Spatial Stream-SVMP+TC	**59.4%**	**57.9%**	**86.6%**	**88.9%**
Temporal Stream-AP [59, 62]	55.2%	60.0%	86.3%	87.2%
Temporal Stream-MP	54.8%	58.5%	86.5%	86.1%
Temporal Stream-SVMP	61.8%	65.7%	88.2%	89.8%
Temporal Stream-SVMP(E2E)	58.3%	63.2%	87.1%	87.8%
Temporal Stream-SVMP+TC	**62.6%**	**67.1%**	**88.8%**	**90.9%**
Two-Stream-AP [59, 62]	58.2%	63.8%	90.6%	91.8%
Two-Stream-MP	56.7%	60.6%	90.1%	87.4%
Two-Stream-SVMP	66.1%	71.0%	92.2%	94.2%
Two-Stream-SVMP(E2E)	63.5%	68.4%	90.6%	92.3%
Two-Stream-SVMP+TC	**67.2%**	**71.3%**	**92.5%**	**94.8%**

SVMP for Action Anticipation Apart from applying SVMP on entire sequences, we also evaluated the usefulness of SVMP for the task of action anticipation. This is motivated by the intuition that SVMP might be able to learn generalizable decision boundaries when shown only a small part of the sequence – given the SVM is optimized in a max-margin framework. Specifically, we only use $k \times \frac{1}{5}$ initial part of the sequences to be pooled by SVMP, ($k \in \{1,2,3,4,5\}$) which has to now predict the action in the full segment. We use the ResNet feature for this experiment. The results are provided in Table 4.5 and is clear that compared with standard pooling methods, SVMP is better, when only seeing a small fraction of the data, supporting our intuition.

SVMP Image: In Figure 4.9, we visualize SVMP descriptor when applied directly on raw video frames. We compare the resulting image against those from other schemes such as the dynamic images of [13]. It is clear that SVMP captures the essence of action dynamics in more detail. To understand the action information present in these images, we trained an action classifier directly on these images, as is done on Dynamic images in [13]. We use the BVLC CaffeNet [97] as the CNN – same the one used in [13]. The results are shown in the Table 4.6 on split-1 of JHMDB (a subset of HMDB-51, containing 21 classes) and UCF-101.

Table 4.5 Comparison of action anticipation on split-1 in UCF-101 and HMDB-51.

HMDB-51					
k/5	1/5	2/5	3/5	4/5	1
SVMP	**58.3%**	**65.5%**	**68.4%**	**70.1%**	**71.0%**
AP	48.6%	56.4%	59.9%	62.5%	63.8%
MP	46.2%	55.4%	56.3%	58.8%	60.6%
UCF-101					
SVMP	**88.3%**	**90.2%**	**93.5%**	**94.0%**	**94.2%**
AP	78.9%	82.4%	87.9%	89.6%	91.8%
MP	79.0%	81.5%	85.6%	86.3%	87.4%

Table 4.6 Recognition rates on split-1 of JHMDB and UCF-101.

Datasets	JHMDB	UCF-101
Mean image	31.3%	52.6%
Max image	28.6%	48.0%
Dynamic image [13]	35.8%	57.2%
SVMP image	**45.8%**	**65.4%**

As is clear, SVMP images are seen to outperform [13] by a significant margin, suggesting that SVMP captures more discriminative and useful action-related features. However, we note that in contrast to dynamic images, our SVMP images do not intuitively look like motion images; this is perhaps because our scheme captures different information related to the actions, and we do not use smoothing (via running average) when generating them. The use of random noise features as the negative bag may be adding additional artifacts.

4.4.4 Experiments on Hand-crafted Features:

As alluded to above, we use the MPII Cooking activities dataset for analyzing the influence of SVMP on hand-crafted features. We use the 4000-D bag-of-words encoded HOG and dense trajctories as our pooling features. In Table 4.7, we report results from this experiment. As is clear, our scheme shows superior performance against the best prior methods (such as rank pooling [64]) on this dataset, showing that our scheme is agnostic to feature type.

Table 4.7 Recognition rates on MPII Cooking dataset.

Method	HOG	Trajectory
AP	45.0%	42.1%
RankPool [64]	47.8%	51.2%
SVMP	**55.0%**	**53.2%**

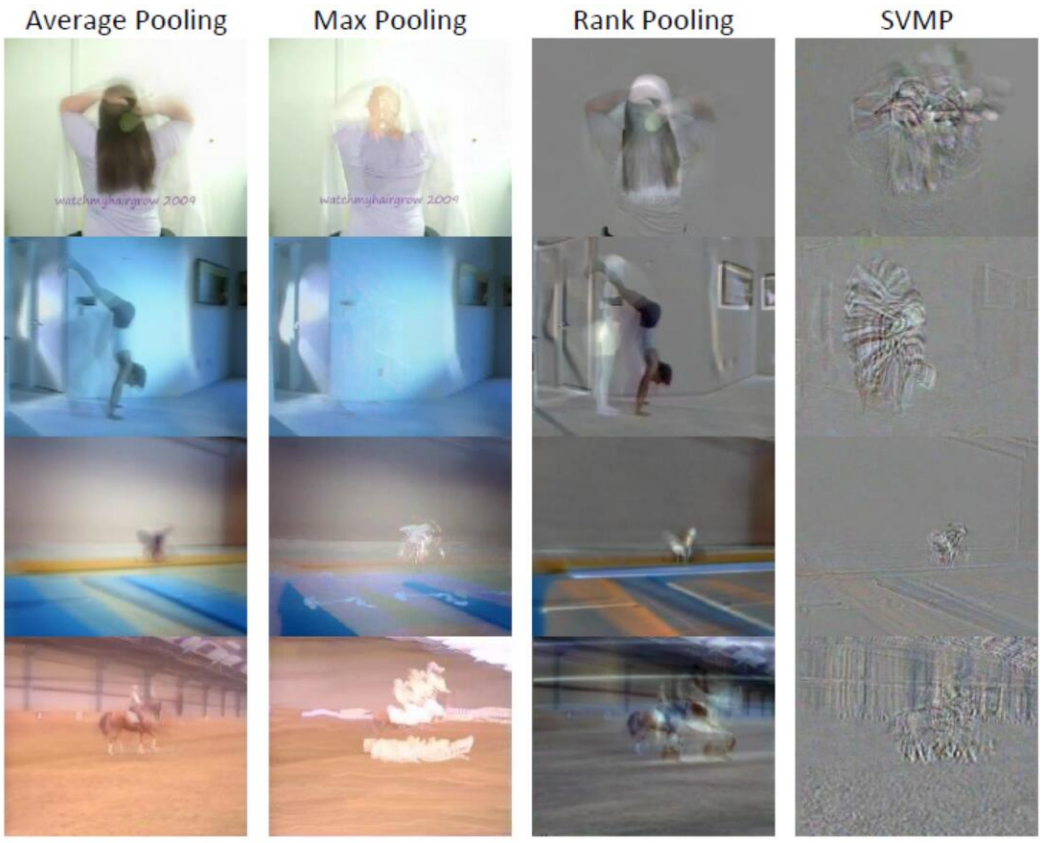


Fig. 4.9 Visualizations of various pooled descriptors.

Table 4.8 Comparisons on Kinetics-600 dataset using I3D feature.

Method	Accuracy
AP [22]	71.9%
MP	67.8%
SVMP	**73.5%**

4.4.5 Action Recognition at Large Scale

Kinetics-600 is one the largest state-of-the-art dataset for action recognition on trimmed videos. For this experiment, we use the I3D network [24] (using the Inception-V3 architecture), as the baseline for feature generator. This model is pre-trained on ImageNet dataset [106] and stacks 64 continuous frames as inputs. Specifically, we extract the CNN features from the second last layer (Mix5c) and apply average pooling to reshape the feature from 4 x 7 x 7 x 1024 into 1024-D vector for each 64-chunk of RGB frames. For each video clip, we use a sliding window to generate a sequence of such features with a window size of 64 and a temporal stride of 8 frames. Then, we apply our proposed SVMP to generate video descriptors for action recognition. In Table 4.8, we make comparisons with the baseline result on the validation set of Kinetics-600, and indicates that SVMP can bring clear improvements even on the large-scale setting.

4.4.6 Action Recognition/Detection in untrimmed videos

We ues the Charades untrimmed dataset for this task. We use the publicly available two-stream VGG features from the fc7 layer for this dataset. We trained our models on the provided training set (7985 videos), and report results (mAP) on the provided validation set (1863 videos) for the tasks of action classification and detection. In the classification task, we concatenate the two-stream features and apply a sliding window pooling scheme to create multiple descriptors. Following the evaluation protocol in [168], we use the output probability of the classifier to be the score of the sequence. In the detection task, we consider the evaluation method with post-processing proposed in [167], which uses the averaged prediction score of a temporal window around each temporal pivots. Instead of average pooling, we apply the SVMP. From Table 4.9, it is clear that SVMP improves performance against other pooling schemes by a significant margin; the reason for this is perhaps the following. During training, we use trimmed video clips, however, when testing, we extract features from every frame/clip in the untrimmed test video. As the network has seen only action-related frames during training, features from background frames may result in arbitrary predictions; and average pooling or max pooling on those features would hurt performance.

Table 4.9 Comparisons on Charades dataset.

Tasks	AP	MP	SVMP
Classification (mAP)	14.2%	15.3	**26.3%**
Detection (mAP)	10.9%	9.2	**15.1%**

When optimizing the binary classification problem between positive and negative bags for SVMP, the decision boundary would capture the most discriminative data support, leading to better summary of the useful features and leading to improved performance.

4.4.7 SVMP Evaluation on Other Tasks

In this section, we provide comprehensive evaluations justifying the usefulness of SVMP on non-video datasets and non-action tasks. We consider experiments on images sets recognition, skeleton-sequence based action recognition, and dynamic texture understanding.

MSR Action3D: In this experiment, we explore the usefulness of SVMP on non-linear geometric features. Specifically, we chose the scheme of Vemulapilli et al. [197] as the baseline that generates Lie algebra based skeleton encodings for action recognition. While they resort to a dynamic time warping kernel for the subsequent encoded skeleton pooling, we propose to use SVMP instead. We use the random noise with the dataset mean and deviation as the negative bag, which achieve better performance.

NTU-RGBD: On this dataset, we apply our SVMP scheme on the skeleton-based CNN features. Specifically, we use [104] as the baseline, which applies a temporal CNN with residual connections on the vectorized 3D skeleton data. We swap the global average pooling layer in [104] by SVM pooling layer. For the evaluation, we adopt the official cross-view and cross-subject protocols. What's interesting here is we try to explore whether the dimension of the feature point would affect the SVMP performance. During the SVMP, we use feature points with dimension from 150 to 4096. It seems only the number of data points would affect the performance of SVMP (from Charades dataset experiment), and it is not sensitive for the dimensionality.

PubFig: In this task, we evaluate the use of SVMP for image set representation. We follow the evaluation setting in [82] and create the descriptor for the training and testing by applying SVMP over ResFace-101 [128] features from every image in the PubFig dataset. Unlike the video-based tasks, all input features in this setting are useful and represent the same person; however their styles vary significantly, which implies the CNN features may be very different even if they are from the same person. This further demands that SVM pooling

would need to find discriminative dimensions in the features that are correlated and invariant to the person identity.

YUP++: To investigate our SVMP scheme on deeper architectures, we use features from the latest Inception-ResNet-v2 model [183], which has achieved the state-of-the-art performance on the 2015 ILSVRC challenge. Specifically, we extract the RGB frames from videos and divide them into training and testing split according to the setting in [61] (using a 10/90 train test ratio). Like the standard image-based CNNs, the clip level label is used to train the network on every frame.

4.4.8 Comparisons to the State of the Art

In Table 4.10, we compare our best results against the state-of-the-art on each dataset in 2018 using the standard evaluation protocols. For a fair comparison, we also report on SVMP combined with hand-crafted features (IDT-FV) [199] for HMDB-51. Our scheme outperforms other methods on all datasets by 1–4%. For example, on HMDB-51, our results are about 2-3% better than the next best method without IDT-FV. On Charades, we outperform previous methods by about 3% while faring well on the detection task against [167]. We also demonstrate significant performance (about 3-4%) improvement on NTU-RGBD and marginally better performance on MSR datasets on skeleton-based action recognition. Our results are superior (by 1-2%) on the PubFig and YUP++ datasets.

We further analyze the benefits of combining I3D+ with SVMP (instead of their proposed average pooling) on both HMDB-51 and UCF-101 datasets using the settings in [24]. However, we find that the improvement over average pooling in I3D+ is not significant; which we believe is because learning the SVMP descriptor needs to solve a learning problem implicitly, requiring sufficient number of training samples, i.e., number of frames in the sequence. The I3D network uses 64-frame chunks as one sample, thereby reducing the number of samples for SVMP, leading to sub-optimal learning. We analyze this hypothesis in Table 4.11; each column in this table represents performances on a data subset, filtered as per the minimum number of frames in their sequences. As is clear from the table, while SVMP performs on par with I3D+ when the sequences are shorter, it demonstrates significant benefits on subsets having longer sequences.

4.5 Chapter Summary

In this chapter, we presented a simple, efficient, and powerful pooling scheme – SVM pooling – for video representation learning. We cast the pooling problem in a multiple instance

Table 4.10 Comparison to the state of the art in each dataset, following the official evaluation protocol for each dataset.

HMDB-51 & UCF-101 (accuracy over 3 splits)		
Method	HMDB-51	UCF-101
Temporal segment networks[211]	69.4%	94.2%
AdaScan[100]	54.9%	89.4%
AdaScan + IDT + C3D[100]	66.9%	93.2%
ST ResNet[59]	66.4%	93.4%
ST ResNet + IDT[59]	70.3%	94.6%
ST Multiplier Network[60]	68.9%	94.2%
ST Multiplier Network + IDT[60]	72.2%	94.9%
Hierarchical rank pooling[63]	65.0%	90.7%
Two-stream I3D[24]	66.4%	93.4%
Two-stream I3D+ (Kinetics 300k)[24]	80.7%	98.0%
Ours (SVMP)	71.3%	94.6%
Ours (SVMP+IDT)	**72.6**%	**95.0**%
Ours (I3D+)	**81.8**%	**98.5**%
Kinetics-600		
Method	Accuracy	
I3D RGB[22]	71.3%	
Second-order Pooling [33]	54.7%	
Ours (SVMP)	**73.5%**	
Charades (mAP)		
Method	Classification	Detection
Two-stream[169]	14.3%	10.9%
ActionVlad + IDT[71]	21.0%	-
Asynchronous Temporal Fields [167]	22.4%	12.8%
Ours (SVMP)	26.3%	15.1%
Ours (SVMP+IDT)	**27.4%**	**16.3%**
MPII Cooking Activity		
Method	Accuracy	
RankPool [64]	51.2%	
Ours (SVMP)	**55.0%**	
MSR-Action3D		
Method	Accuracy	
Lie Group[197]	92.5%	
ST-LSTM + Trust Gate[122]	94.8%	
Ours (SVMP)	**95.5%**	
NTU-RGBD		
Method	Cross-Subject	Cross-View
Res-TCN[104]	74.3%	83.1%
ST-LSTM + Trust Gate[122]	69.2%	77.7%
Ours (SVMP)	**79.4%**	**87.6%**
PubFig		
Method	Accuracy	
Deep Reconstruction Models[82]	89.9%	
ESBC[83]	98.6%	
Ours (SVMP)	**99.3%**	
YUP++		
Method	Stationary	Moving
Temporal Residual Networks[61]	92.4%	81.5%
Ours (SVMP)	**92.9%**	**84.0%**

Table 4.11 Accuracy comparison on different subsets of HMDB-51(H) and UCF-101(U) split-1 using I3D+ features.

Min # of frames	1	80	140	180	260
# of classes (H)	51	49	27	21	12
# of classes (U)	101	101	95	82	52
I3D (H)	79.6%	81.8%	84.1%	78.0%	77.3%
SVMP (H)	**80.0%**	**82.9%**	**84.8%**	**85.1%**	**86.8%**
I3D (U)	98.0%	98.0%	98.0%	95.9%	93.8%
SVMP (U)	**98.4%**	**98.9%**	**99.3%**	**98.5%**	**97.3%**

learning framework, and seek to learn useful decision boundaries on video features against background/noise features. We provide an efficient scheme that jointly learns these decision boundaries and the action classifiers on them. Extensive experiments were showcased on eight challenging benchmark datasets, demonstrating state-of-the-art performance. Given the challenging nature of these datasets, we believe the benefits afforded by our scheme is a significant step towards the advancement of recognition systems designed to represent sets of images or videos. As is introduced in the Section 4.1.1, the decision boundary of SVM can be treated as the weighted average of support vectors. This shares the similar inspiration with self-attention mechanism [195] that are widely used in the transformer based architecture [52, 11, 4, 207]. While our method learns attention over support vectors inside SVM, self-attention in transformer generate attention weighting through computing similarity between non-overlapped image patches. Compared to our method in this thesis, vision transformers operate over low-level image data which enables it to build both local and global dependencies spatio-temporally but with higher cost.

Chapter 5

Contrastive Video Representation Learning via Adversarial Perturbations

Deep learning has enabled significant advancements in several areas of computer vision; however, the sub-area of video-based recognition continues to be elusive. In comparison to image data, the volumetric nature of video data makes it significantly more difficult to design models that can remain within the limitations of existing hardware and the available training datasets. In Chapter 3 and 4, we introduce ordered pooling and discriminative pooling scheme to capture temporal order and discriminative information respectively, which help to get better video representations. However, there are two unanswered concerns that merit more investigation, 1), how to effectively get the noise pattern in the discriminative pooling? 2), Whether or not one decision boundary is enough for capturing the discriminative video data distribution?

In this Chapter, we present a novel pooling framework to contrastively summarize the temporally-ordered video features. Different from prior works, we assume that per-frame video features consist of noisy parts that could confuse a classifier in a downstream task, such as for example, action recognition. A robust representation, in this setting, will be one that could avoid the classifier from using these vulnerable features for making predictions. Learning such representations is similar in motivation to the idea of contrastive learning [76], which has achieved promising results in many recent works for the task of unsupervised visual feature learning [6, 87, 89, 140, 185, 218, 234, 84, 28]. These works learn the visual representation by contrasting the positive pairs against the negative ones via a loss function, namely as contrastive loss. However, our work in this chapter is fundamentally different from previous contrastive learning, as we do not cast it in the unsupervised learning. We start from pre-trained model for extracting visual features. By applying our proposed algorithm, we aim to improve the video representation. In previous works, it is challenging to generate negative

samples for video sequences and building the memory bank is also not appealing. To this end, we resort to some intuitions made in a few works recently in the area of adversarial perturbations [125, 139, 133, 219]. Such perturbations are noise-like patterns that, when added to data, can fail an otherwise well-trained highly accurate classifier. Such perturbations are usually subtle, and in image recognition tasks, are quasi-imperceptible to a human. It was shown in several recent works that such noise can be learned from data. Specifically, by taking gradient ascent on a minimizing learning objective, one can produce such perturbations that will push the data points to the class boundaries, thereby making the classifier to mis-classify. Given that the strength (norm) of this noise is often bounded, it is highly likely that such noise will find minimum strength patterns that select features that are most susceptible to mis-classification. To this end, we use the recent universal adversarial perturbation generation scheme [133].

Once the perturbations are learned (and fixed) for the dataset, we use it to learn robust representations for the video. To this end, for features from every frame, we make two bags, one consisting of the original features, while the other one consisting of features perturbed by noise. Next, we learn a discriminative hyperplane that separates the bags in a max-margin framework. Such a hyperplane, which in our case is produced by a primal support vector machine (SVM), finds decision boundaries that could well-separate the bags; the resulting hyperplane is a single vector and is a weighted combination of all the data points in the bags. Given that the data features are non-linear, and given that a kernelized SVM might not scale well with sequence lengths, we propose to instead use multiple hyperplanes for the classification task, by stacking several such hyperplanes into a column matrix. We propose to use this matrix as our data representation for the video sequence.

However, there is a practical problem with our descriptor; each such descriptor is local to its respective sequences and thus may not be comparable between videos. To this end, we make additional restrictions on the hyperplanes – regularizing them to be orthogonal, resulting in our representation being subspaces. Such subspaces mathematically belong to the so-called Stiefel manifold [15]. We formulate a novel objective on this manifold for learning such subspaces on video features. Further, as each feature is not independent of the previous ones, we make additional temporal constraints. We provide efficient Riemannian optimization algorithms for solving our objective, specifically using the Riemannian conjugate gradient scheme that has been used in several other recent works [31, 78, 93]. Our overall pipeline is graphically illustrated in Figure 5.1.

We present experiments on three video recognition tasks, namely (i) action recognition, (ii) dynamic texture recognition, and (iii) 3D skeleton based action recognition. On all the

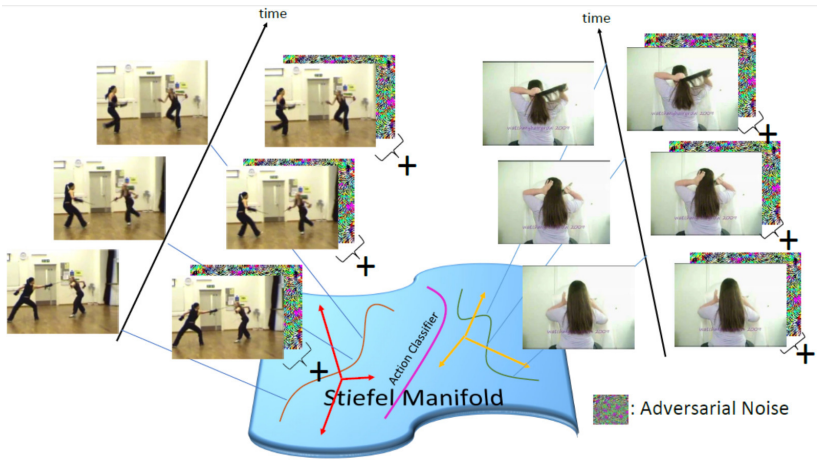


Fig. 5.1 A graphical illustration of our discriminative subspace pooling with adversarial noise. For every video sequence (as CNN features), our scheme generates a positive bag (with these features) and a negative bag by adding adversarial perturbations to the features. Next, we learn discriminative temporally-ordered hyperplanes that separate the two bags. We use orthogonality constraints on these hyperplanes and use them as representations for the video. As such representations belong to a Stiefel manifold, we use a classifier on this manifold for video recognition.

experiments, we show that our scheme leads to better results, often improving the accuracy between 3–14% compared with our baseline methods.

5.1 Related Work

5.1.1 Contrastive Learning

One way to improve the data representation is via contrastive learning [76], which learns representations by minimizing a contrastive loss between the positive and negative pairs. This approach has been used in several recent works [75, 6, 87, 89, 140, 185, 218, 234, 84, 28], achieving promising results for unsupervised visual representation learning. Although their motivations are different, the core idea is to unsupervisely train an encoder by minimizing the contrastive loss, which encodes a visual representation closer to its positive data points and far away from its negatives. The difference with our formulation is that some works [6, 87, 89, 140, 185, 234, 28] formulate the positive and negative pairs within a mini-batch while others [84, 218] build a memory bank for generating the pairs. However, as the video data is often voluminous, it is hard to apply the same strategy for learning video representations. Moreover, the size of the memory bank could be huge due to the potentially

large spatio-temporal semantic complexity in the video data. Even though some works avoid using memory bank or generating negative pairs [75, 30, 29], they are still under the category of self-supervise learning, which requires large-scaled dataset to produce better feature representation on general purpose. Instead, we show how we can use adversarial perturbations [133] to produce negative samples that benefits the downstream task, in a network-agnostic manner, which can then be used for contrastive learning within a novel subspace-based contrastive learning framework. Specifically, different from the classic contrastive learning methods mentioned above, we formulate a binary classification problem that contrasts the video features against its perturbed counterparts and use the learned decision boundaries as video representation.

5.1.2 Adversarial Perturbation

Our main inspiration comes from the recent work of Moosavi et al. [133] that show the existence of quasi-imperceptible image perturbations that can fool a well-trained CNN model. They provide a systematic procedure to learn such perturbations in an image-agnostic way. In Xie et al. [219], such perturbations are used to improve the robustness of an object detection system. Similar ideas have been explored in [125, 139, 230]. In Sun et al. [181], a latent model is used to explicitly localize discriminative video segments. In Chang et al. [27], a semantic pooling scheme is introduced for localizing events in untrimmed videos. In Miyato et al. [131], virtual adversarial perturbation is generated to improve the feature representation learning in weakly supervised set-up. While these schemes share similar motivation as ours, the problem setup and formulations are entirely different.

5.2 Approach

Let us assume $X = \langle x_1, x_2, ..., x_n \rangle$ be a sequence of video features, where $x_i \in \mathbb{R}^d$ represents the feature from the i-th frame. We use 'frame' in a loose sense; it could mean a single RGB frame or a sequence of a few RGB or optical flow frames (as in the two stream [171] or the I3D architectures [24]) or a 3D skeleton. The feature representation x_i could be the outputs from intermediate layers of a CNN. As alluded to in the introduction, our key idea is the following. We look forward to an effective representation of X that is (i) compact, (ii) preserves characteristics that are beneficial for the downstream task (such as video dynamics), and (iii) efficient to compute. Recent methods such as generalized rank pooling [31] have similar motivations and propose a formulation that learns compact temporal descriptors that are closer to the original data in ℓ_2 norm. However, such a reconstructive objective may also

capture noise, thus leading to sub-optimal performance. Instead, we take a different approach in the contrastive learning fashion. Specifically, we assume to have access to some noise features $Z = \{z_1, z_2, ..., z_m\}$, each $z_i \in \mathbb{R}^d$. Let us call X the positive bag, with a label $y = +1$ and Z the negative bag with label $y = -1$. Our main goal is to find a discriminative hyperplane that separates the two bags; these hyperplanes can then be used as the representation for the bags.

An obvious question is how such a hyperplane can be a good data representation? To answer this, let us consider the following standard SVM formulation with a single discriminator $w \in \mathbb{R}^d$:

$$\min_{w,\xi \geq 0} \frac{1}{2} \|w\|^2 + \sum_{\theta \in X \cup Z} \left[\max(0, 1 - y(\theta) w^\top \theta + \xi_\theta) + C\xi_\theta \right], \tag{5.1}$$

where with a slight abuse of notation, we assume $y(\theta) \in \{+1, -1\}$ is the label of θ, ξ are the slack variables, and C is a regularization constant on the slacks. Given the positive and negative bags, the above objective learns a linear classification boundary that could separate the two bags with a classification accuracy of say γ. If the two bags are easily separable, then the number of support vectors used to construct the separating hyperplane might be a few and thus may not capture a weighted combination of a majority of the points in the bags — as a result, the learned hyperplane would not be representative of the bags. However, if the negative bag Z is suitably selected and we demand a high γ, we may turn (5.1) into a difficult optimization problem and would demand the solver to overfit the decision boundary to the bags; this overfitting creates a significantly better summarized representation, as it may need to span a larger portion of the bags to satisfy the γ accuracy.[1] This overfitting of the hyperplane is our key idea, that allows to avoid using data features that are susceptible to perturbations, while summarizing the rest.

There are two key challenges to be addressed in developing such a representation, namely (i) an appropriate noise distribution for the negative bag, and (ii) a formulation to learn the separating hyperplanes. We explore and address these challenges below.

5.2.1 Finding Noise Patterns

As alluded to above, having good noise distributions that help us identify the vulnerable parts of the feature space is important for our scheme to perform well. The classic contrastive learning schemes either pair random noise [185, 206] or use in-batch samples [28, 84] as the negatives, in which the random noise may introduce uncertainty into the learned feature

[1] Here regularization parameter C is mainly assumed to help avoid outliers.

Input: Feature points x_{ij}, Network weighting W, fooling rate ψ, cross entropy loss with softmax funtion $f(.)$, normalization operator $N(.)$.
Output: Adversarial noise vector ε.
Initialization: $\varepsilon \leftarrow 0$.
repeat
  $\Delta\varepsilon \leftarrow \arg\min_r \|r\|_2 - \sum_{ij} f(W^\top(x_{ij}), W^\top(x_{ij} + \varepsilon + r))$;
  $\varepsilon \leftarrow N(\varepsilon + \Delta\varepsilon)$;
until $Accuracy \leq 1 - \psi$;
return v

Algorithm 3: Optimization step for solving adversarial noise.

and the in-batch samples would take too much memory during the training of video data. Instead, we resort to the recent idea of universal adversarial perturbations (UAP) [133] to formulate the negatives by adding this global noise pattern onto the positives. This scheme is dataset-agnostic and provides a systematic and mathematically grounded formulation for generating adversarial noise that when added to the original features is highly-likely to mis-classify a pre-trained classifier. Further, this scheme is computationally efficient and requires less data for building relatively generalizable universal perturbations.

Precisely, suppose $\mathscr{X}$ denotes our dataset, let h be a CNN trained on $\mathscr{X}$ such that $h(x)$ for $x \in \mathscr{X}$ is a class label predicted by h. Universal perturbations are noise vectors ε found by solving the following objective:

$$\min_{\varepsilon} \|\varepsilon\| \text{ s.t. } h(x+\varepsilon) \neq h(x), \forall x \in \mathscr{X}, \tag{5.2}$$

where $\|\varepsilon\|$ is a suitable normalization on ε such that its magnitude remains small, and thus will not change x significantly. In [133], it is argued that this norm-bound restricts the optimization problem in (5.2) to look for the minimal perturbation ε that will move the data points towards the class boundaries; i.e., selecting features that are most vulnerable – which is precisely the type of noise we need in our representation learning framework.

To this end, we extend the scheme described in [133], to our contrastive setting. Differently to their work, we aim to learn a UAP on high-level CNN features as detailed in Alg. 3 above, where the x_{ij} refers to the i^{th} frame in the j^{th} video. We use the classification accuracy before and after adding the noise as our optimization criteria as captured by maximizing the cross-entropy loss.

5.2.2 Discriminative Subspace Pooling

Once a "challenging" noise distribution is chosen, the next step is to find a summarization technique for the given video features. While one could use a simple discriminative classifier, such as described in (5.1) to achieve this, such a linear classifier might not be sufficiently powerful to separate the potentially non-linear CNN features and their perturbed counterpart. An alternative is to resort to non-linear decision boundaries using a kernelized SVM; however that may make our approach less scalable and poses challenges for end-to-end learning. Thus, we look forward to a representation within the span of data features, while having more capacity for separating non-linear features.

Our main idea is to use a subspace of discriminative directions (as against a single one as in (5.1)) for separating the two bags such that every feature x_i is classified by at least one of the hyperplanes to the correct class label. Such a scheme can be looked upon as an approximation to a non-linear decision boundary by a set of linear ones, each one separating portions of the data. Mathematically, suppose $W \in \mathbb{R}^{d\times p}$ is a matrix with each hyperplane as its columns, then we seek to optimize:

$$\min_{W,\xi} \Omega(W) + \sum_{\theta\in X\cup Z} \left[\max\left(0, 1 - \max\left(\mathbf{y}(\theta) \odot \mathbf{W}^\top \theta\right) - \xi_\theta\right) + C\xi_\theta\right], \tag{5.3}$$

where $\mathbf{y}$ is a vector with the label y repeated p times along its rows. The quantity Ω is a suitable regularization for W, of which one possibility is to use $\Omega(W) = W^\top W = \mathbf{I}_p$, in which case W spans a p dimensional subspace of $\mathbb{R}^d$. Enforcing such subspace constraints (orthonormality) on these hyperplanes are often empirically seen to demonstrate better performance as is also observed in [31]. The operator $\odot$ is the element-wise multiplication and the quantity $\max(\mathbf{y}(\theta) \odot \mathbf{W}^\top \theta)$ captures the maximum value of the element-wise multiplication, signifying that if at least one hyperplane classifies θ correctly, then the hinge-loss will be zero.

Recall that we work with video data, and thus there are temporal characteristics of this data modality that may need to be captured by our representation. In fact, recent works show that such temporal ordering constraints indeed results in better performance, e.g., in action recognition [31, 64, 13, 12]. However, one well-known issue with such ordered pooling techniques is that they impose a global temporal order on all frames jointly. Such holistic ordering ignores the repetitive nature of human actions, for example, in actions such as clapping or hand-waving. As a result, it may lead the pooled descriptor to overfit to non-repetitive features in the video data, which might be corresponding to noise/background. Usually a slack variable is introduced in the optimization to handle such repetitions, however its effectiveness is questionable. To this end, we propose a simple temporal segmentation

based ordering constraints, where we first segment a video sequence into multiple non-overlapping temporal segments $\mathscr{T}_0, \mathscr{T}_1, ...\mathscr{T}_{\lfloor n/\delta \rfloor}$, and then enforce ordering constraints only within the segments. We find the segment length δ as the minimum number of consecutive frames that do not result in a repeat in the action features.

With the subspace constraints on W and introducing temporal segment-based ordering constraints on the video features, our complete **order-constrained discriminative subspace pooling optimization** can be written as:

$$\min_{\substack{W^\top W=\mathbf{I}_p,\\ \xi,\zeta\geq 0}} \sum_{\theta\in X\cup Z}\Big[\max\Big(0,1-\max\Big(\mathbf{y}(\theta)\odot \mathbf{W}^\top\theta\Big)-\xi_\theta\Big)\Big]+C_1\sum_{\theta\in X\cup Z}\xi_\theta+C_2\sum_{i<j}\zeta_{ij}, \tag{5.4}$$

$$\left\|W^\top x_i\right\|^2+1\leq\left\|W^\top x_j\right\|^2+\zeta_{ij},\quad i<j,\forall(i,j)\in\mathscr{T}_k, \text{where} \tag{5.5}$$

$$\mathscr{T}_k=\{k\delta+1,k\delta+2,...,\min(n,(k+1)\delta)\},\forall k\in\{0,1,...,\lfloor n/\delta\rfloor\} \tag{5.6}$$

$$\delta=b^*-a^*,\ \text{where}\ (a^*,b^*)=\underset{a,b>a}{\arg\min}\left\|x_a-x_b\right\|, \tag{5.7}$$

where (5.5) captures the temporal order, while the last two equations define the temporal segments, and computes the appropriate segment length δ, respectively. Note that, the temporal segmentation part could be done offline, by using all videos in the dataset, and selecting a δ which is the mean. In the next section, we present a scheme for optimizing W by solving the objective in (5.4)and (5.5).

Once each video sequence is encoded by a subspace descriptor, we use a classifier on the Stiefel manifold for recognition. Specifically, we use the standard exponential projection metric kernel [31, 79] to capture the similarity between two such representations, which are then classified using a kernelized SVM.

5.2.3 Efficient Optimization

The orthogonality constraints on W results in a non-convex optimization problem that may seem difficult to solve at first glance. However, note that such subspaces belong to well-studied objects in differential geometry. Specifically, they are elements of the Stiefel manifold $\mathscr{S}(d,p)$ (p subspaces in $\mathbb{R}^d$), which are a type of Riemannian manifolds with positive curvature [15]. There exists several well-known optimization techniques for solving objectives defined on this manifold [3], one efficient scheme is Riemannian conjugate gradient (RCG) [173]. This method is similar to the conjugate gradient scheme in Euclidean spaces, except that in the case of curved-manifold-valued objects, the gradients should adhere to the geometry (curvature) of the manifold (such as orthogonal columns in our case), which can be achieved via suitable projection operations (called exponential maps). However,

such projections may be costly. Fortunately, there are well-known approximate projection methods, termed *retractions* that could achieve these projections efficiently without losing on the accuracy. Thus, tying up all together, for using RCG on our problem, the only part that we need to derive is the Euclidean gradient of our objective with respect to W. To this end, rewriting (5.5) as a hinge loss on (5.4), our objective on W and its gradient are:

$$\min_{W\in\mathscr{S}(d,p)} g(W) := \sum_{\theta\in X\cup Z}\left[\max\left(0,1-\max\left(\mathbf{y}(\theta)\odot\mathbf{W}^\top\theta\right)-\xi_\theta\right)\right] + \frac{1}{n(n-1)}\sum_{i<j}\max(0,1+\left\|W^\top x_i\right\|^2-\left\|W^\top x_j\right\|^2-\zeta_{ij}), \tag{5.8}$$

$$\frac{\partial g}{\partial W} = \sum_{\theta\in X\cup Z} A(W;\theta,y(\theta)) + \frac{1}{n(n-1)}\sum_{i<j} B(W;x_i,x_j), \text{where} \tag{5.9}$$

$$A(W;\theta,y(\theta)) = \begin{cases} 0, & \text{if } \max(y(\theta)\odot W^\top\theta-\xi_\theta)\geq 1 \\ -\left[\mathbf{0}_{d\times r-1}\ y(\theta)\theta\ \mathbf{0}_{d\times p-r}\right],\ r=\arg\max_q y(\theta)\odot W_q^\top\theta, & \text{else} \end{cases} \tag{5.10}$$

$$B(W;x_i,x_j) = \begin{cases} 0, & \text{if } \left\|W^\top x_j\right\|^2\geq 1+\left\|W^\top x_i\right\|^2-\zeta_{ij} \\ 2(x_ix_i^\top - x_jx_j^\top)W, & \text{else.} \end{cases} \tag{5.11}$$

In the definition of $A(W)$, we use $W_q^\top$ to denote the q-th column of W. To reduce clutter in the derivations, we have avoided including the terms using $\mathscr{T}$. Assuming the matrices of the form xx^T can be computed offline, on careful scrutiny we see that the cost of gradient computations on each data pair is only $O(d^2p)$ for $B(W)$ and $O(dp)$ for the discriminative part $A(W)$. If we include temporal segmentation with k segments, the complexity for $B(W)$ is $O(d^2p/k)$.

5.3 End-to-End CNN Learning

End-to-end CNN training through the discriminative subspace pooling (DSP) layer can be done using methods that are quite well-known. For a reader who might be unfamiliar with such methods, we provide a detailed exposition below. To set the stage for our discussions, we first provide our CNN architecture with the DSP layer. This CNN model is depicted in Figure 5.2. In the model, we assume the DSP layer takes as input the feature map X_{L-1} from the previous layer (across all frames) and the adversarial noise Z, and produces as output the subspace descriptor W^*. This W^* goes through another series of CNN fully connected layers before using it in a loss layer $\mathscr{L}$ (such as cross-entropy) to be trained against a ground truth video class c. Among the gradients of parameters S on the various blocks, the only non-trivial

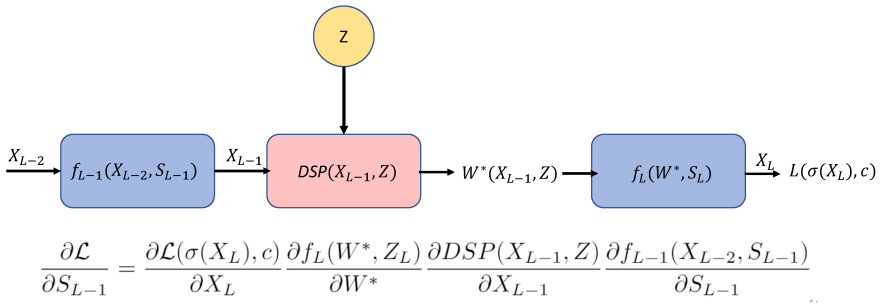


Fig. 5.2 Architecture of our end-to-end CNN with discriminative subspace pooling (DSP) layer in between. We assume X_ℓ represents the feature map outputs from the ℓ-th CNN layer (from all frames in the sequence) denoted as f_ℓ, and S_ℓ represents its respective parameters. The final loss is shows as $\mathscr{L}$, $\sigma(\beta)$ is the softmax function, and c is the action class label. The parameter W is the subspace pooled output of the DSP layer, and Z is the adversarial noise. Below the model, we provide the gradient that we are after for enabling back-propagation through the DSP layer.

gradient is the one for the block penultimate to the DSP layer, to update the parameters S_{L-1} of this layer will require the gradient of the DSP block with respect to its inputs X_{L-1} (the gradient that we are interested in is depicted below our CNN model in Figure 5.2). The main challenge to have this gradient is that it is not with regard to the weights W, but the outcome of the DSP optimization W^* – which is an argmin problem, that is:

$$W^* = \arg\min_W \mathrm{DSP}(X_{L-1}, Z). \tag{5.12}$$

Given that the Riemannian objective might not be practically amenable to a CNN setup (due to its components such as exponential maps, etc. that might be expensive in a CNN setting), we use a slightly different objective in this setup, given below (which is a variant of Eq. 5.3 in Section 5.2.2). We avoid the use of the ordering constraints in our formulation, to simplify our notations (however we use it in our experiments).

$$\min_W \mathrm{DSP}(X) := \Omega(W) + \sum_{i=1}^{n} \left[\max\left(0, 1 - \max\left(y_i W^\top X^i\right)\right)\right]^2, \tag{5.13}$$

where $\Omega(W) = \left\|W^T W - \mathbf{I}_p\right\|_F^2$ is the subspace constraint specified as a regularization. Recall that y_i is binary label for frame i. With a slight abuse of notation to avoid the proliferation of the CNN layer L in the derivations, we use X to consist of both the data features and the adversarial noise features, as captured by their labels in y ($y = -1$ for adversarial noise

features and 1 otherwise), and that the pair (X^i, y_i) denote the i-th column of X and its binary label respectively.

5.3.1 Gradients for Argmin

In this section, we derive the gradient $\frac{\partial \text{DSP}(X)}{\partial X}$. We use the following theorem for this derivation, which is well-known as the implicit function theorem [37], [58][Chapter 5] and recently reviewed in Gould et al. [74].

Theorem 1. *Let* $\text{DSP} : \mathbb{R}^{d\times n} \to \mathbb{R}^{d\times p}$ *be our discriminative subspace pooling operator on n features each of dimension d (defined as in* (5.13)*). Then, its gradient wrt* X^i *is given by:*

$$\nabla_{X^i} \text{DSP}(W;X) = -\left\{\nabla_{WW} \text{DSP}(W;X)\right\}^{-1} \nabla_{X^iW} \text{DSP}(W;X^i)\Big|_{W=W^*} \tag{5.14}$$

The above theorem suggests that to get the required gradient, we only need to find the second derivatives of our objective. To simplify notation, let $P(t,q)$ denote a $d \times p$ matrix, with all zeros, except the q-th column which is t. Then, for all i satisfying $\max(y_i W^T X^i) < 1$, we have the second-order derivatives as follows:

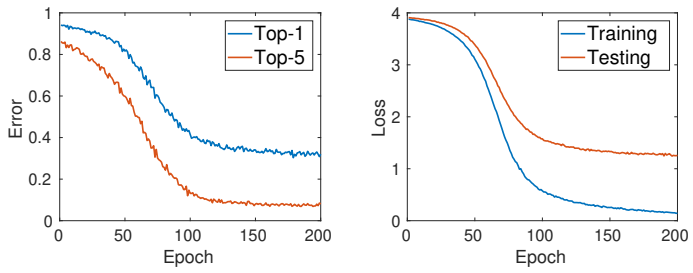
$$\nabla_{WW} \text{DSP}(W;X) = \Omega''(W) + 2\sum_i \text{vec}\left(P\left(\alpha_j(i), j(i)\right)\right) \text{vec}\left(P\left(\alpha_j(i), j(i)\right)\right)^T, \tag{5.15}$$

where $j(i) = \arg\max_q y_i W^T X^i$ and $\alpha_j(i) = y_i X^i$, q capturing the dimension-index that takes the largest of $y_i W^T X^i$, which is a $p \times 1$ vector. Similarly,

$$\nabla_{X^iW} \text{DSP}(W;X) = 2\,\text{vec}\left(P\left(\alpha_j(i), j(i)\right)\right) \text{vec}\left(P\left(\beta, j(i)\right)\right)^T, \tag{5.16}$$

where j and α_j are as defined above, while $\beta = P(y_i W, j(i))$. Note that ∇_{WW} is a $pd \times pd$ matrix, while ∇_{X^iW} is a $pd \times d$ matrix. While, it may seem expensive to compute these large matrices, note that it requires only the vectors $y_i X^i$ as its elements which are cheap to compute, and the argmin takes only linear time in p, which is quite small (6 in our experiments). However, computing the matrix inverse of ∇_{WW} can still be costly. To avoid this, we use a diagonal approximation to it.

Figures 5.3(a) and 5.3(b) show the convergence and the action classification error in the end-to-end learning setup on the HMDB-51 dataset split1 using a ResNet-152 model.



(a) (b)

Fig. 5.3 Convergence of our end-to-end training setup on HMDB-51 split1.

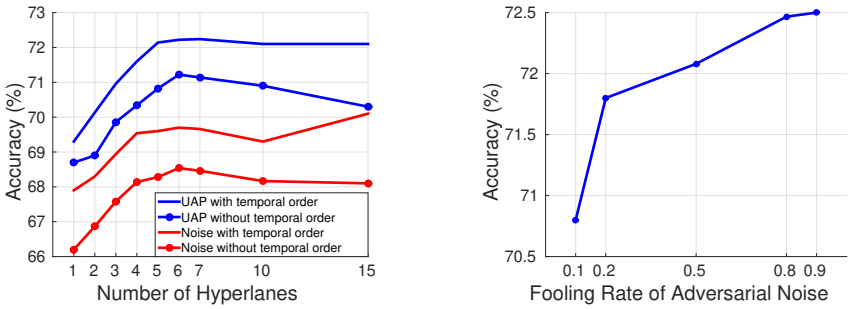
5.4 Experiments

In this section, we demonstrate the utility of our discriminative subspace pooling (DSP) on several standard vision tasks (including action recognition, skeleton-based video classification, and dynamic video understanding), and on diverse CNN architectures such as ResNet-152, Temporal Convolutional Network (TCN), and Inception-ResNet-v2. We implement our pooling scheme using the ManOpt Matlab package [17] and use the RCG optimizer with the Hestenes-Stiefel's [77] update rule. We found that the optimization produces useful representations in about 50 iterations and takes about 5 milli-seconds per frame on a single core 2.6GHz CPU. We explore different values of the slack regularization constant C and finally set its value as 1. As for the CNN features, we used public code for the respective architectures to extract the features. Generating the adversarial perturbation plays a key role in our algorithm, as it is used to generate our negative bag for learning the discriminative hyperplanes. We follow the experimental setting in [133] to generate UAP noise for each model by solving the energy function as depicted in Alg. 3. Differently from [133], we generate the perturbation in the shape of the high level CNN feature instead of an RGB image. We review below our the datasets, their evaluation protocols, the CNN features next.

5.4.1 Data Preprocessing

HMDB-51 [107]: To extract features, we train a two-stream ResNet-152 model (as in [170]) taking as input RGB frames (in the spatial stream) and a stack of optical flow frames (in the temporal stream). We use features from the pool5 layer of each stream as input to DSP, which are sequences of 2048D vectors.

NTU-RGBD [165]: We use the scheme in Shahroudy et al. [165] as our baseline in which a temporal CNN (with residual units) is applied on the raw skeleton data. We use the 256D



(a) (b)

Fig. 5.4 Analysis of the hyper parameters used in our scheme. All experiments use ResNet-152 features on HMDB-51 split-1 with a fooling rate of 0.8 in (a) and 6 hyperplanes in (b). See text for details.

features from the bottleneck layer (before their global average pooling layer) as input to our scheme.

YUP++ dataset [61]: We train an Inception-ResNet-v2 on the respective training set to generate the features and fine-tune a network that was pre-trained on the ImageNet dataset. In detail, we apply the 1/9 train-test ratio and follow the standard supervised training procedure of image-based tasks; following which we extract frame-level features (1536D) from the second-last fully-connected layer.

5.4.2 Parameter Analysis

Evaluating the Choice of Noise: As is clear by now, the noise patterns should be properly chosen in the contrastive learning setup, as it will affect how well the discriminative hyperplanes characterize useful video features. To investigate the quality of UAP features, we compare it with the baseline of choosing noise from a Gaussian distribution with the data mean and standard deviation computed on the respective video dataset (as done in the work of Wang et al. [206]). We repeat this experiment 10-times on the HMDB-51 split-1 features. In Figure 5.4(a), we plot the average classification accuracy after our pooling operation against an increasing number of hyperplanes in the subspaces. As is clear, using UAP significantly improves the performance against the alternative, substantiating our intuition. Further, we also find that using more hyperplanes is beneficial, suggesting that adding UAP to the features leads to a non-linear problem requiring more than a single discriminator to capture the informative content.

	HMDB-51			NTU-RGBD		YUP++	
	Spatial	Temporal	Two-stream	CS	CV	Stationary	Moving
AP	46.7%	60.0%	63.8%	74.3%	83.1%	85.1%	76.5%
MP	45.1%	58.5%	60.6%	65.4%	78.5%	81.8%	72.4%
DSP	**58.5%**	**67.0%**	**72.5%**	**81.6%**	**88.7%**	**95.1**%	**88.3**%

Table 5.1 The accuracy comparison between our Discriminative subspace pooling (DSP) with standard Average pooling (AP) and Max pooling (MP), where CS represent Cross-Subject and CV represent Cross-View

Evaluating Temporal Constraints: Next, we evaluate the merit of including temporal-ordering constraints in the DSP objective, viz. Equation (5.5). In Figure 5.4(a), we plot the accuracy with and without such temporal order, using the same settings as in the above experiment. As is clear, embedding temporal constraint will help the discriminative subspace capture representations that are related to the video dynamics, thereby showing better accuracy. In terms of the number of hyperplanes, the accuracy increases about 3% from one hyperplane to when using six hyperplanes, and drops around 0.5% from 6 hyperplanes to 15 hyperplanes, suggesting that the number of hyperplanes (6 in this case) is sufficient for representing most sequences.

UAP Fooling Rate: In Figure 5.4(b), we analyze the fooling rate of UAP that controls the quality of the adversary to confuse the trained classifier. The higher the fooling rate is, the more it will mix the information of the feature in different classes. As would be expected, we see that increasing the fooling rate from 0.1 to 0.9 increases the performance of our pooling scheme as well. Interestingly, our algorithm could perform relatively well without requiring a very high value of the fooling rate. From [133], a lower fooling rate would reduce the amount of data needed for generating the adversarial noise, making their algorithm computationally cheaper. Further, comparing Figures 5.4(a) and 5.4(b), we see that incorporating a UAP noise that has a fooling rate of even 10% does show substantial improvements in DSP performance against using Gaussian random noise (70.8% in Figure 5.4(b) against 69.8% in Figure 5.4(a)).

Experimental Settings: Going by our observations in the above analysis, for all the experiments in the sequel, we use six subspaces in our pooling scheme, use temporal ordering constraints in our objective, and use a fooling rate of 0.8 in UAP. Further, as mentioned earlier, we use an exponential projection metric kernel [35] for the final classification of the subspace descriptors using a kernel SVM.

5.4.3 Experimental Results

Compared with standard pooling: In Table 5.1, we show the performance of DSP on the three datasets and compare to standard pooling methods such as average pooling and max pooling. As is clear, we outperform the baseline results by a large margin. Specifically, we achieve 9% improvement on the HMDB-51 dataset split-1 and $5\% - 8\%$ improvement on the NTU-RGBD dataset. On these two datasets, we simply apply our pooling method on the CNN features extracted from the pre-trained model. We achieve a substantial boost (of up to 12%) after applying our scheme.

Comparisons to the State of the Art: In Table 5.2, we compare DSP to the state-of-the-art results on each dataset. On the HMDB-51 dataset, we also report accuracy when DSP is combined hand-crafted features (computed using dense trajectories [199] and summarized as Fisher vectors (IDT-FV)). As the results show, our scheme achieves significant improvements over the state of the art. For example, without IDT-FV, our scheme is 3% better than than the next best scheme [211] (69.4% vs. 72.4% ours). Incorporating IDT-FV improves this to 74.3% which is again better than other schemes. We note that the I3D architecture [24] was introduced recently that is pre-trained on the larger Kinectics dataset and when fine-tuned on the HMDB-51 leads to about 80.9% accuracy. To understand the advantages of DSP on pooling I3D model generated features, we applied our scheme to their bottleneck features (extracted using the public code provided by the authors) from the fine-tuned model. We find that our scheme further improves I3D by about 0.6% showing that there is still room for improvement for this model. On the other two datasets, NTU-RGBD and YUP++, we find that our scheme leads to about 5–7% and 3–6% improvements respectively, and outperforms prior schemes based on recurrent networks and temporal relation models, suggesting that our pooling scheme captures spatio-temporal cues much better than recurrent models.

Run Time Analysis: In Figure 5.5, we compare the run time of DSP with similar methods such as rank pooling, dynamic images, and GRP. We used the Matlab implementations of other schemes and used the same hardware platform (2.6GHz Intel CPU single core) for our comparisons. To be fair, we used a single hyperplane in DSP. As the plot shows, our scheme is similar in computations to rank pooling and GRP.

Analysis of Results on I3D Features: To understand why the improvement of DSP on I3D (80.9% against our 81.5%) is not significant (on HMDB-51) in comparison to our results on other datasets, we further explored the reasons. Apparently, the I3D scheme uses chunks of 64 frames as input to generate one feature output. However, to obtain DSP representations, we

HMDB-51	
Method	Accuracy
Temporal Seg. n/w [211]	69.4%
TS I3D [24]	80.9%
ST-ResNet [59]	66.4%
ST-ResNet+IDT [59]	70.3%
STM Network [60]	68.9%
STM Network+IDT [60]	72.2%
ShuttleNet+MIFS [166]	71.7%
GRP [31]	70.9%
SVMP [206]	71.0%
L^2STM [182]	66.2%
Ours(TS ResNet)	**72.4%**
Ours(TS ResNet+IDT)	**74.3%**
Ours(TS I3D)	**81.5%**

NTU-RGBD		
Method	Cross-Subject	Cross-View
VA-LSTM [231]	79.4%	87.6%
TS-LSTM [112]	74.6%	81.3%
ST-LSTM+Trust Gate [122]	69.2%	77.7%
SVMP [206]	78.5%	86.4%
GRP [31]	76.0%	85.1%
Res-TCN [177]	74.3%	83.1%
Ours	**81.6%**	**88.7%**
YUP++		
Method	Stationary	Moving
TRN [61]	92.4%	81.5%
SVMP [206]	92.5%	83.1%
GRP [31]	92.9%	83.6%
Ours	**95.1%**	**88.3%**

Table 5.2 Comparisons to the state-of-the-art on each dataset following their respective official evaluation protocols. We used three splits for HMDB-51. 'TS' refers to 'Two-Stream'.

#frames	1	80	100	140	160	180	260
#classes	51	49	34	27	23	21	12
AP [24]	**80.8**	81.8	86.1	84.1	82.3	78.0	**77.3**
DSP (ours)	**81.6**	82.8	88.5	88.0	86.1	83.3	**82.6**

Table 5.3 Comparison of I3D performance on sequences of increasing lengths in HMDB-51 split-1.

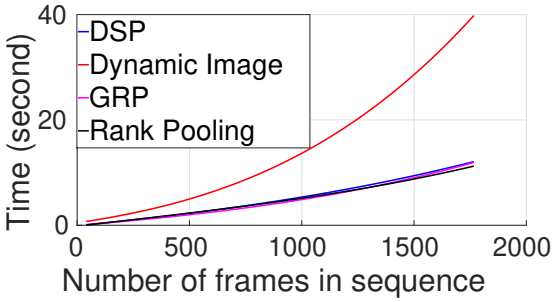


Fig. 5.5 Run time analysis of DSP against GRP [31], RP [64], and Dynamic Images [13]

need a sufficient number of features per video sequence to solve the underlying Riemannian optimization problem adequately, which may be unavailable for shorter video clips. To this end, we re-categorized HMDB-51 into subsets of sequences according to their lengths. In Table 5.3, we show the performance on these subsets and the number of action classes for sequences in these subsets. As our results show, while the difference between average pool (AP) (as is done in [24]) and DSP is less significant when the sequences are smaller ($<$80 frames), it becomes significant ($>$5%) when the videos are longer ($>$260 frames). This clearly shows that DSP on I3D is significantly better than AP on I3D.

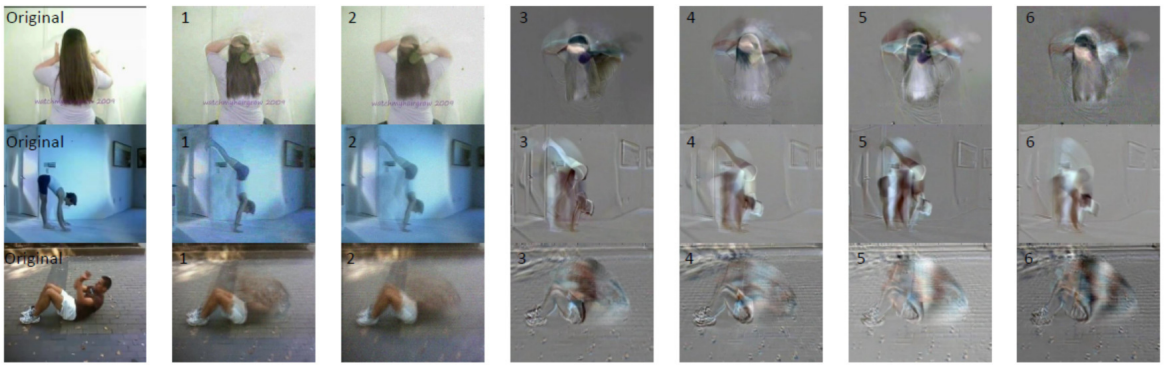


Fig. 5.6 Visualizations of our DSP descriptor (when applied on raw RGB frames) on an HMDB-51 video sequences. First column shows a sample frame from the video, second-to-seventh columns show the six hyperplanes produced by DSP. Interestingly, we find that each hyperplane captures different aspects of the sequences–first two mostly capture spatial, while the rest capture the temporal dynamics at increasing granularities.

Qualitative Results: In Figure 5.6, we visualize the hyperplanes that our scheme produces when applied to raw RGB frames from HMDB-51 videos – i.e., instead of CNN features, we directly feed the raw RGB frames into our DSP, with adversarial noise generated as suggested

in [133]. We find that the subspaces capture spatial and temporal properties of the data separately; e.g., the first two hyperplanes seem to capture mostly the spatial cues in the video (such as the objects, background, etc.) while the rest capture mostly the temporal dynamics at greater granularities. Note that we do not provide any specific criteria to achieve this behavior, instead the scheme automatically seem to learn such hyperplanes corresponding to various levels of discriminative information.

5.5 Chapter Summary

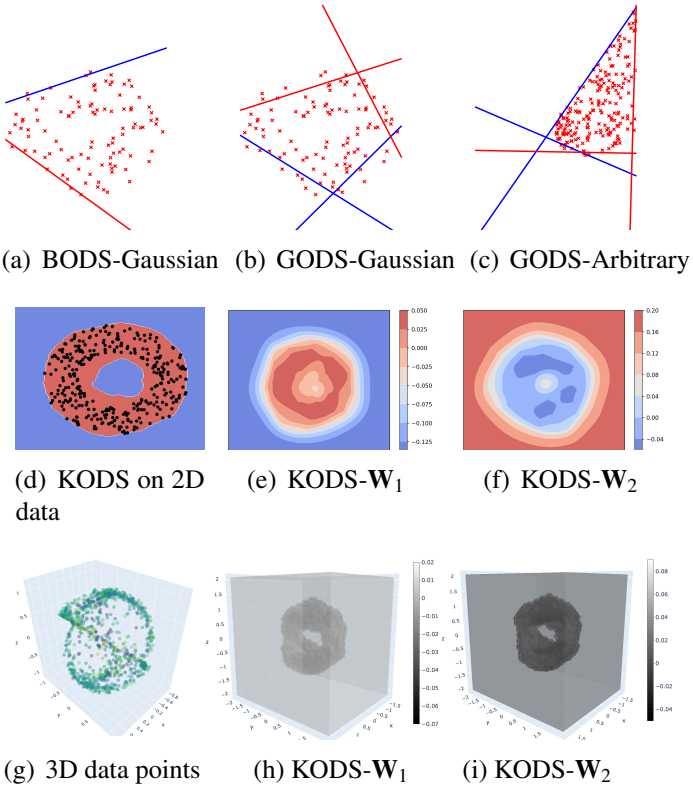
In this Chapter, we investigated the problem of contrastive representation learning for video sequences. Our main innovation is to generate and use synthetic noise, in the form of adversarial perturbations, for building the negative pairs, and then producing our video representation in a novel contrastive pooling scheme. Assuming the video frames are encoded as CNN features, such perturbations are often seen to affect vulnerable parts of the features. Using such generated perturbations to our benefit, we propose a discriminative classifier, in a max-margin set-up, via learning a set of hyperplanes as a subspace, that could separate the data from its perturbed counterpart. As such hyperplanes need to fit to useful parts of the features for achieving good performance, it is reasonable to assume they capture data parts that are robust. We provided a non-linear objective for learning our subspace representation and explored efficient optimization schemes for computing it. Experiments on several datasets explored the effectiveness of each component in our scheme, demonstrating state-of-the-art performance on the benchmarks. In this Chapter, we learn the video representation by contrasting between frame-level representation and their perturbated counterpart. Recently, many popular contrastive learning works [75, 84, 29] have the similar motivation with ours. While our work defines a binary classification problem in weakly supervised learning fashion, these works evolved in another direction. They implement the contrastive learning in unsupervised manner and define positive pairs as the representation from the same sample with different argumentations and negative pairs as the representation from different samples. Then, they force the representation of positive pairs as similar as possible and the representation of negative pairs as dissimilar as possible. Compared to our work in this thesis, these works aim at general feature representation learning, conditioned on large-scaled training dataset.

Chapter 6

One-Class Video Representation Learning Using Pairs of Complementary Classifiers

In earlier Chapters, we discussed several video representation learning techniques for aggregating frame-level features into video-level representations while preserving the video sequence's temporal ordering and discriminative information. In the Discriminative Pooling (Chapter 4) and Discriminative Subspace Pooling (Chapter 5), the decision boundary from a binary classifier is used as the video representation. After that, we use this representation to do multi-class classification task. The experimental results indicate that the decision boundary from classifier effectively capture the discriminative data distribution by applying proper constraint in objectives. In this Chapter, we are going to extend this feature learning method to another video-related task, namely One-Class video representation learning. Unlike the problem set-up before, only one class of videos will be given, and they are labelled as positive. The task is to learn the characteristic of positives and detect both positives and non-positives (negatives) during the inference.

Classical solutions to one-class problems are based on support vector machines (SVMs), such as the one-class SVM (OC-SVM), that maximizes the margin of the discriminative hyperplane from the origin [164]. There are extensions of this scheme, such as the least-squares one-class SVM (LS-OSVM) [40] or its online variants [212], that learn to find a tube of minimal diameter that includes all the labeled data. Another popular approach is the support-vector data description (SVDD) that finds a hypersphere of minimum radius that encapsulates the training data [184]. There have also been kernelized extensions of these schemes that use the kernel trick to embed the data points in a reproducible kernel Hilbert space, potentially enclosing the 'normal' data with arbitrarily-shaped boundaries.



(a) BODS-Gaussian (b) GODS-Gaussian (c) GODS-Arbitrary

(d) KODS on 2D data (e) KODS-$\mathbf{W}_1$ (f) KODS-$\mathbf{W}_2$

(g) 3D data points (h) KODS-$\mathbf{W}_1$ (i) KODS-$\mathbf{W}_2$

Fig. 6.1 Visualizations of decision regions using various GODS formulations on synthetic data. Figures (a, b, c) show subspaces found by basic one-class discriminative subspaces (BODS) and generalized one-class discriminative subspaces (GODS) on various data distributions (see Section 6.5.5). The colors identify hyperplanes within a classifier in the complementary pair. Figure (d) shows KODS decision regions for ring-shaped data (black dots) using an RBF kernel. Figures (e, f) are the decision regions of the classifiers; $\mathbf{W}_1$ bounding data from outside and $\mathbf{W}_2$ from inside, together they define the region in (d). Figure (g,h,i) show 3D points and the decision surfaces of the two classifiers.

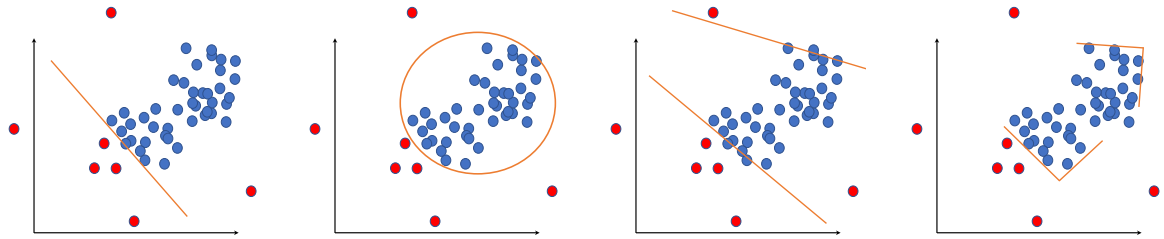
Apart from the classic one-class solutions, there is an increasing number of recent works that use deep neural networks for building the one-class model [1, 25, 157, 123, 127, 80]. In these approaches, typically a deep auto-encoder model is trained on the one-class data such that its reconstruction error is minimized. When such a model is provided with an out-of-distribution data sample (anomaly), the reconstruction error can be large, which could be used as an anomaly cue [80, 157]. There are extensions of this general architecture using generative adversarial networks (GANs) to characterize the distribution of the in-class samples [163, 151]. For anomaly detection on time-varying inputs, there are also adaptations using predictive auto-encoders, such as [123], that attempts to generate the (latent) future

samples, and flags anomalies if the predicted sample is significantly different from the observed one.

While, these approaches have been widely adopted in several applications (see e.g., Chandala et al. [26]), they have drawbacks. For example, the OC-SVM uses only a single hyperplane, however using multiple hyperplanes may be beneficial [201]. The SVDD scheme makes a strong assumption on the spherical nature of the data distribution. Using kernel methods may impact scalability, while deep learning methods may need specialized hardware (such as GPUs) and large training sets. Thus, trading-off between the pros and cons of these diverse prior methods, we propose novel generalizations of these techniques, which we call *generalized one-class discriminative subspaces* (GODS). The key goal of GODS is to combine the linearity properties of OC-SVM, and the non-linear bounded characterization of SVDD in a single framework. However, our proposed one-class model is neither linear nor uses a spherical classifier, instead uses a pair of orthornormal frames[1] whose columns characterize linear classifiers. Specifically, these columns are optimized such that the one-class data belongs to the positive half-spaces of the columns in one of these classifiers and to the negative half-spaces of the columns in the other; thus these classifiers jointly form a complementary pair. These classifiers, with their respective half-spaces defined by their orthonormal columns, non-linearly bound the data from different directions. Our learning objective, to find these complementary classifiers, jointly optimizes two opposing criteria: i) to minimize the distance between the two classifiers, thus bounding the data within the smallest volume, and ii) to maximize the margin between the hyperplanes and the data, thereby avoiding overfitting, while improving classification robustness.

Our proposed GODS model offers several advantages against prior methods: (i) the piecewise linear decision boundaries approximate a non-linear classifier, while providing computationally cheap inference, and (ii) the use of the complementary classifier pair allows flexible bounding the data distribution of arbitrary shapes, as illustrated in Figure 6.1. For example, our kernelized variant of GODS, that we introduce in Section 6.2.4, bounds data from outside as well as inside (see Figure 6.1(d)), which is usually not possible in prior methods. Albeit these benefits, our objective is non-convex due to the orthogonality constraints. However, such non-convexity fortunately is not a significant practical concern as they naturally place the optimization objective on the Stiefel manifold [55]. This is a well-studied Riemannian manifold [15] for which there exist efficient non-linear optimization methods at our disposal. We use one such optimization scheme, dubbed Riemannian conjugate gradient [3], which is fast and efficient.

[1] Orthonormal frames are matrices with linearly independent unit norm columns.



(a) OC-SVM (b) SVDD (c) BODS (ours) (d) GODS (ours)

Fig. 6.2 A graphical illustration of OC-SVM and SVDD in relation to our proposed BODS and GODS schemes. The blue points show the given one-class data, the red points are outliers (which are not available at training), and the decision boundaries are shown by orange curves/lines.

To empirically evaluate the benefits of our GODS formulations, we apply them to one-class data arising from various anomaly detection problems in computer vision and machine learning. One novel task we consider is that of out-of-pose (OOP) detection in cars [191, 190]. Specifically, in this task, our goal is to detect if the passengers or the driver are seated OOP as captured by an inward looking dashboard camera. For this task, we showcase the effectiveness of our approaches on a new dataset, which we call Dash-Cam-Pose. Apart from this task, we also report experimental results on several standard and public anomaly detection benchmarks in computer vision, such as on the UCF-crime [180] and the UCSD Ped2 [116] datasets. We also provide experiments on the standard JHMDB action recognition dataset [96] re-purposed for the anomaly detection task. We further analyse the generalizability of our approach to non-computer vision applications by providing results on five UCI datasets. Our results demonstrate that GODS variants lead to significant performance improvements over the baseline methods.

6.1 Related work

6.1.1 one-class classification

As one-class problems arise in numerous practical settings, they have been explored to great depth in a variety of disciplines, including but not limited to remote sensing [129], network intruder detection [110], and fraud detection [86]. In computer vision, a few illustrative problems are novelty detection [68, 102, 157], video anomaly detection [47, 148, 160], diagnosis on medical images [105], and anomalous object attribute recognition in image collections [159]. For a comprehensive review of applications, we refer the interested reader to excellent surveys, such as [26, 147].

Classic methods for modelling such one-class problems are extensions of data density estimation techniques [192, 136]. These methods attempt to model the density of the given data in the input space by trading-off between maximizing their inclusivity within a (given) density quantile while minimizing its volume. The dependence on minimal density volume in the input space is discarded in Scholkopf et al. [164, 184] against smoothness of the decision function in a non-linear (kernelized) feature space. Working in the kernel space not only allows for more flexible characterizations of the distribution of the data samples, but also allows transferring the max-margin machinery (and the associated theory) developed for support vector machines to be directly used in the one-class setting. However, as alluded to earlier, working with kernel feature maps can be demanding in large data settings, and thus our main focus in this chapter is on deriving one-class algorithms in the input space. That said, we also explore a kernelized dual variant of our scheme for problems that are impossible to be modelled using our primal variant.

Modern efforts to one-class learning typically use either (i) good hand-crafted representations combined with effective statistical learning models, or (ii) implicit representation and model learning via neural works. Below, we review these efforts in detail.

Explicit Modelling Approaches. Performance of any one-class approach inevitably depends on the effectiveness of the data representation. For example, visual representations such as histogram of oriented gradients [45] (HOG) and histogram of optical flows (HOF) [46] have been beneficial in developing several anomaly detection algorithms. A Markov random filed (MRF) on HOG and HOF descriptors is proposed in Zhang et al. [228] for modeling the normal patterns in a semi-supervised manner, where an abnormal sample model is iteratively derived from the normal one using Bayesian adaptation. In Xu and Caramanis [222], an outlier pursuit algorithm is proposed using convex optimization for the robust PCA problem. Similarly, Kim et al. [103] propose a space-time MRF to detect abnormal activities in videos. This method uses a mixture of probabilistic principal component analysis to characterize the distribution of normal data characterized as densities on optical flow. Detecting out-of-context objects is explored in [39, 141] using support graph and generative models. Motion trajectory analysis [217, 20, 193] of objects in video sequences has been a common approach for modeling anomalies, under the strong assumption that deviant trajectories may correspond to abnormal data. Detecting salient regions in images has also been implemented by some researchers [94, 99]. In contrast to these approaches that propose problem-specific anomaly detection models, our solution is for a general setting.

Sparse reconstruction analysis has been a powerful workhorse in the recent times in developing several one-class solutions [43, 124, 232, 114]. The assumption in these methods is that normal data can be encoded as sparse linear combinations of columns in a dictionary

that is learned only on the normal data; however the reconstruction error of any out-of-distribution sample using this dictionary could be significant. In addition to the reconstruction loss, a study from Ren et al. [152] points out that the sparsity term should be taken into consideration for improving the anomaly detection accuracy. However, sparse reconstruction methods can be computationally expensive. To improve their efficiency, Bin et al. [232] proposes an online detection scheme using sparse reconstructibility of query signals from an atomically learned event dictionary. Yang et al. [43] improves efficiency via learning multiple small dictionaries to encode image patterns at multiple scales. While, our proposed GODS algorithm could also be treated as a dictionary of orthonormal columns, our inference is significantly cheaper in contrast to solving ℓ_1-regularized problems in these works as GODS involves only evaluations of inner products of the data samples to the learned hyperplanes.

Deep Learning Based Methods. The huge success of deep neural networks on several fundamental problems in computer vision [48, 72] has also casted its impact in devising schemes for anomaly detection [80]. Extending classical methods, a deep variant of SVDD is proposed in Ruff et al. [155], however assumes the one-class data is unimodal. Variants of OC-SVM are explored in [25, 221]. Parera and Patel [145] proposes a trade-off between compactness and descriptiveness using an external reference dataset to train a deep model on one-class data. Liang et al. [120] proposes to use statistical trends in the softmax predictions. Deep learning based feature representations have been used as replacements for hand-crafted features in several one-class problem settings. For example, Xu et al. [220] design a multi-layer auto-encoder embracing data-driven feature learning. Similarly, Hasan et al. [80] propose a 3D convolutional auto-encoder to capture both spatial and temporal cues in video anomaly detection. Leveraging the success of convolutional neural networks (CNNs) to capture spatial cues, [41, 126], and [127] propose to embed recurrent networks, such as LSTMs, to model the appearance dynamics of normal data. In [113] and [121], frameworks to minimize the in-distribution sample distances is proposed thereby maximizing the distance to out-of-distribution samples. In Sabokrou et al. [156], a pre-trained CNN is used for extracting region features, and a cascaded outlier detection scheme is applied. Multiple instance learning (MIL) in a deep learning setting is attempted in [180] for anomaly detection using weakly-labeled training videos via applying an MIL ranking loss with sparsity and smoothness constraints; however includes both normal and abnormal samples in the training set. Deep generative adversarial networks (GAN) have also been proposed to characterize the single class [150, 157, 163, 151]. These methods typically follow the same philosophy of training auto-encoders, however uses an adversarial discriminator to improve quality of the decoded data sample; the discriminator confidence is then used as an abnormality cue during inference.

In contrast to these approaches, we focus on explicit modelling of one-class data distributions, allowing better and more controlled characterization of the single-class. Deep learning approaches reviewed above are complimentary to our contributions; in fact we use deep-learned data representations in our experiments and simultaneously demonstrate our performances on non-deep-learned features as well.

6.2 Approach

In this section, we formally introduce our schemes. First, we present BODS using a pair of hyperplanes, which we generalize to GODS using a pair of discriminative frames in Section 6.2.2. We explore variants of GODS in Section 6.2.3 and further generalize GODS using kernel feature maps, proposing KODS in Section 6.2.4. With a slight abuse of terminology, we call our entire suite of formulations as GODS.

6.2.1 Basic One-class Discriminative Subspaces

In this section, we introduce a basic variant of our objective, which we call Basic One-class Discriminative Subspaces (BODS). The key goal of BODS is to bound the one-class data distribution using a pair of hyperplanes. Similar to OC-SVM, we seek these hyperplanes to have a maximum margin from the data distribution, thus allowing robustness to minor differences in the test data distribution. Further, inspired by SVDD, we also demand the two hyperplanes to bound the data within a minimal data region. BODS combines these two conflicting objectives into a joint formulation. Mathematically, suppose $(\mathbf{w}_1, b_1)$ and $(\mathbf{w}_2, b_2)$ define the parameters of the pair of hyperplanes. Our goal in BODS is then to minimize an objective such that all data points $\mathbf{x}_i$ be classified to the positive half-space of $(\mathbf{w}_1, b_1)$ and to the negative half-space of $(\mathbf{w}_2, b_2)$, while also minimizing a suitable distance between the two hyperplanes. To this end, we propose the following BODS objective:

$$\min_{\substack{(\mathbf{w}_1,b_1),(\mathbf{w}_2,b_2),\\ \xi_1,\xi_2,\beta>0}} \frac{1}{2}\mathrm{dist}^2((\mathbf{w}_1,b_1),(\mathbf{w}_2,b_2)) + \boldsymbol{\Omega}(\xi_{1i},\xi_{2i}), \tag{6.1}$$

$$\text{subject to } \left(\mathbf{w}_1^T\mathbf{x}_i + b_1\right) \geq \eta - \xi_{1i}, \tag{6.2}$$

$$\left(\mathbf{w}_2^T\mathbf{x}_i + b_2\right) \leq -\eta + \xi_{2i}, \quad \forall i = 1,2,...,n, \tag{6.3}$$

where (6.2) and (6.3) capture the complementary constraints. We use the notation $\boldsymbol{\Omega}(\xi_{1i},\xi_{2i}) = C\sum_{i=1}^{n}\left(\xi_{1i}^2 + \xi_{2i}^2\right)$ for the slack regularization and $\eta > 0$ specifies a (given)

classification margin. The two hyperplanes have their own parameters, however are pulled together by the first term in (6.1) that aims to minimize the distance dist between them. For BODS, we assume dist is the Euclidean distance, i.e., $\text{dist}^2((\mathbf{w}_1, b_1), (\mathbf{w}_2, b_2)) = \|\mathbf{w}_1 - \mathbf{w}_2\|^2 + (b_1 - b_2)^2$.

It is a common practice in machine learning and computer vision applications to pre-process the data features to have unit norm, and is often found to result in superior performances, especially when using deep-learned features. Following this idea, we assume that our data is unit normalized, i.e., $\|\mathbf{x}_i\| = 1$, and thus belongs to a unit hypersphere U^{d-1}, which is a sub-manifold of the Euclidean manifold $\mathbb{R}^d$. This assumption on the data naturally places our hyperplanes also to belong to U^{d-1}; i.e., $\|\mathbf{w}_1\| = \|\mathbf{w}_2\| = 1$. Using these manifold constraints, our BODS formulation can be rewritten as follows:

$$P_1 := \min_{\mathbf{w}_1, \mathbf{w}_2 \in U^{d-1}, b_1, b_2} \frac{1}{2}\boldsymbol{\alpha}(b_1, b_2) - \mathbf{w}_1^T \mathbf{w}_2 \\ + \frac{\nu}{2n} \sum_i \left[\eta - \left(\mathbf{w}_1^T \mathbf{x}_i + b_1\right)\right]_+^2 + \left[\eta + \left(\mathbf{w}_2^T \mathbf{x}_i + b_2\right)\right]_+^2, \tag{6.4}$$

where $\boldsymbol{\alpha}(b_1, b_2) = (b_1 - b_2)^2$. We further simplify the BODS objective by substituting the constraints on the slacks ξ's in (6.2) and (6.3) into $\boldsymbol{\Omega}$ in (6.1) and include them in the objective as soft constraints using the hinge loss $[\ ]_+$. We use ν to denote a penalty factor on these soft constraints. In Fig. 6.2(c), we illustrate the decision boundaries of BODS model.

While, BODS offers a simple and flexible model to capture the one-class distribution, the linearity of the classifiers may limit its applications to sophisticated data models. A natural idea is then to empower these classifiers to have non-linear decision boundaries. While, using a kernel method is perhaps a standard approach in this regard (which we present subsequently), we first propose to achieve non-linearity via piecewise linear decision boundaries. To this end, we equip each classifier in BODS with a set of hyperplanes; each set forming a complementary pair with the other. The use of piecewise linear decision boundaries makes inference computationally cheap as it requires only $2K$ inner products during inference, assuming K hyperplanes per set. Further, we also avoid the need for computing kernel matrices, allowing for scalability of our approach to larger datasets. However, using sets of hyperplanes brings in the challenge of how to effectively regularize them to avoid overfitting and redundancy. To this end, in the following subsections, we generalize BODS to use pairs of multiple hyperplanes, regularized as orthonormal frames, thus providing a richer and non-linear discriminative setup, and subsequently present other regularizations and kernel embeddings.

6.2.2 One-class Discriminative Subspaces

Let us continue to assume the input data is unit normalized, we will remove this assumption in the next section. Formally, suppose $\mathbf{W}_1, \mathbf{W}_2 \in \mathscr{S}K$ be subspace frames – that is, matrices of dimensions $d \times K$, each with K columns where each column is orthonormal to the rest; i.e., $\mathbf{W}_1^T\mathbf{W}_1 = \mathbf{W}_2^T\mathbf{W}_2 = \mathbf{I}_K$, where $\mathbf{I}_K$ is the $K \times K$ identity matrix (see Fig. 6.2(d)). Such frames belong to the so-called Stiefel manifold, denoted $\mathscr{S}K$, with K d-dimensional subspaces. Note that the orthogonality assumption on the $\mathbf{W}_i$'s is to ensure they capture diverse discriminative directions, leading to better regularization; further also improving their characterization of the data distribution. A direct extension of P_1 then leads to:

$$P_2 := \min_{\mathbf{W}\in\mathscr{S}K,\mathbf{b}} \frac{1}{2}\mathrm{dist}_{\mathrm{W}}^2(\mathbf{W}_1,\mathbf{W}_2) + \boldsymbol{\alpha}(\mathbf{b}_1,\mathbf{b}_2) \tag{6.5}$$

$$+ \frac{\nu}{2n}\sum_i \left[\eta - \min(\mathbf{W}_1^T\mathbf{x}_i + \mathbf{b}_1)\right]_+^2 \tag{6.6}$$

$$+ \frac{\nu}{2n}\sum_i \left[\eta + \max(\mathbf{W}_2^T\mathbf{x}_i + \mathbf{b}_2)\right]_+^2, \tag{6.7}$$

where $\mathrm{dist}_{\mathrm{W}}$ is a suitable distance between subspaces, and $\mathbf{b} \in \mathbb{R}^K$ is a vector of biases, one for each hyperplane. Note that in (6.6) and (6.7), unlike BODS, $\mathbf{W}^T\mathbf{x}_i + \mathbf{b}$ is a K-dimensional vector. Thus, (6.6) says that the minimum value of this vector should be greater than η and (6.7) says that the maximum value of it is less than $-\eta$. To simplify the notation, let us use $\zeta(\mathbf{W},\mathbf{b}) = \boldsymbol{\alpha}(\mathbf{b}_1,\mathbf{b}_2) + (6.6) + (6.7)$. Then, P_2 can be written as follows:

$$P_2' := \min_{\mathbf{W}\in\mathscr{S}K,\mathbf{b}} -\mathrm{Tr}\,\mathbf{W}_1^T\mathbf{W}_2 + \zeta(\mathbf{W},\mathbf{b}). \tag{6.8}$$

The formulation P_2', due to the first term, enforces a *tight coupling* between $\mathbf{W}_1$ and $\mathbf{W}_2$; such a coupling might prevent the frames from freely aligning to the data distribution, resulting in sub-optimal performance. To circumvent this issue, we propose the following work around. Recall that the main motivation to define the distance between the subspaces is so that they sandwich the (one-class) data points compactly. Thus, rather than defining a distance between subspaces directly, one could also use a measure that minimizes the Euclidean distance of each data point from both the hyperplanes; thereby achieving the same effect. Such a distance via the data points will also make the frames loosely coupled. More formally, we propose to redefine $\mathrm{dist}_{\mathrm{W}}^2$ as:

$$\text{dist}^2_{\text{W}}(\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}_1, \mathbf{b}_2 | \mathbf{x}) = \frac{1}{2} \sum_{j=1}^{2} \left\| \mathbf{W}_j^T \mathbf{x} + \mathbf{b}_j \right\|^2, \tag{6.9}$$

where now we minimize the sum of the lengths of each $\mathbf{x}$ after projecting on to the respective frames; thereby pulling both the frames closer to the data point. Using this definition of dist^2_{W}, we formulate our generalized one-class discriminative subspace (GODS) classifier as:

$$\begin{aligned} \text{GODS} := \min_{\mathbf{W} \in \mathscr{S}K, b} F = \frac{1}{2n} \sum_{i=1}^{n} \sum_{j=1}^{2} \left\| \mathbf{W}_j^T \mathbf{x}_i + \mathbf{b}_j \right\|^2 \\ + \frac{\nu}{2n} \sum_i \left[\eta - \min(\mathbf{W}_1^T \mathbf{x}_i + \mathbf{b}_1) \right]_+^2 + \left[\eta + \max(\mathbf{W}_2^T \mathbf{x}_i + \mathbf{b}_2) \right]_+^2 \end{aligned} \tag{6.10}$$

6.2.3 Extensions to GODS Formulation

The technical development of GODS in the previous section assumes the input data is unit normalized, as otherwise the orthornormal frames for discriminating them may not be generally fruitful. Our other important assumption in the previous section – that the hyperplanes in our discriminative decision parameters $\mathbf{W}$ are orthonormal – can be restrictive as well. In this section, we provide extensions of our GODS framework that relax or remove these restrictions. In the following variants, we will assume $\lambda > 0$ to generically denote a penalty on the respective regularization.

Non-Compact Stiefel Manifold

A matrix $\mathbf{W} \in \mathbb{R}^{d \times K}$ with its K columns being linearly independent, however not unit normalized, belongs to the so-called non-compact Stiefel manifold (Absil et al. [3][Chapter 3]), denoted $\mathbb{R}_*^{d \times K}$, which is an open subspace of the Euclidean space $\mathbb{R}^{d \times K}$. One may represent such a manifold as a product manifold between a $d \times K$ Stiefel manifold and a $K \times 1$ Euclidean vector; i.e., $\mathbb{R}_*^{d \times K} = \mathscr{S}K \times \mathbb{R}^K$. For example, if $\mathbf{W} \in \mathbb{R}_*^{d \times K}$, then $\mathbf{W} = \mathbf{Q}\,\text{diag}(\mathbf{r})$, where $\mathbf{Q} \in \mathscr{S}K$, $\mathbf{r} \in \mathbb{R}^K$, and the i-th dimension $\mathbf{r}_i = \|\mathbf{W}_{:,i}\|$, the ℓ_2 norm of the i-th column in $\mathbf{W}$. For non-unit-norm input data, we can extend the GODS formulation in (6.10) using a non-compact Stiefel manifold as:

$$\text{GODS}_N := \min_{(\mathbf{Q}, \mathbf{r}) \in \mathscr{S}K \times \mathbb{R}^K, b} F\left(\mathbf{Q}\,\text{diag}(\mathbf{r}), b\right) + \frac{\lambda}{2} \|\mathbf{r}\|_p, \tag{6.11}$$

where F is the objective in (6.10) and $\lambda > 0$ is a penalty on the ℓ_p-norm regularization over $\mathbf{r}$. Note that in (6.11), for brevity we assume $(\mathbf{Q},\mathbf{r}) = \{(\mathbf{Q}_i,\mathbf{r}_i)\}_{=1}^{2}$, i.e., it is technically a product of two non-compact Stiefel manifolds corresponding to the two discriminative subspaces. There is an additional advantage with the proposed subspace representation – it can allow automatic selection of the number of subspace components one may need. For example, with the ℓ_p regularization on $\mathbf{r}$, some of the dimensions in $\mathbf{r}$ can go to zero (say using $p = 1$), thereby removing the respective subspace component from the final representation.

Oblique Manifold

We may also relax the orthogonality constraints on $\mathbf{W}$, however maintain their unit normality. A set of matrices $\mathscr{OB}_d^K = \left\{\mathbf{W} \in \mathbb{R}^{d\times K} : \text{diag}(\mathbf{W}^\top\mathbf{W}) = \mathbf{I}_K\right\}$ forms a regular submanifold of the Euclidean manifold and is usually called an Oblique manifold [2] under the canonical inner product metric. This manifold is isometric to the product of K spheres $\times_1^K \mathscr{S}1$. We can rewrite a variant of GODS with the optimization on $\mathscr{OB}_d^K$ as:

$$\text{GODS}_O := \min_{\mathbf{W}\in\mathscr{OB}_d^K,b} F(\mathbf{W},b) + \frac{\lambda}{2}\sum_{i=1}^{2}\left\|\mathbf{W}_i^\top\mathbf{W}_i - \mathbf{I}_K\right\|_F^2, \tag{6.12}$$

the last term *softly* controls the correlations among columns in $\mathbf{W}$.

Euclidean Manifold

Removing both the orthogonality and the unit norm constraints on the classifiers and the input data results in our most general form of the GODS formulation, that assumes $\mathbf{W}$ belongs to the Euclidean manifold. We can write such a variant as:

$$\text{GODS}_E := \min_{\mathbf{W}\in\mathbb{R}^{d\times K},b} F(\mathbf{W},b) + \frac{\lambda}{2}\sum_{i=1}^{2}\left\|\mathbf{W}_i^\top\mathbf{W}_i - \mathbf{I}_K\right\|_F^2. \tag{6.13}$$

Similar to (6.12), the last term in GODS_E controls the correlations between the columns in $\mathbf{W}$; a large λ will promote $\mathbf{W}$ to be similar to the original GODS formulation using the Stiefel manifold in (6.10).

6.2.4 Kernelized One-class Discriminative Subspaces

While, a subspace-based classifier as in our GODS formulation can offer computationally efficient, yet non-linear decision functions in the input space, it may fail in situations when input data cannot be bounded using rectilinear coordinates. A few illustrative examples for

such catastrophic cases could be when the distribution has isolated components (or islands), is xor- or ring-shaped, etc. This lends a kernelized variant of GODS scheme inevitable.

To derive the kernelized GODS, we use our formulation[2] in P_2, however expand the min and max constraints in (6.6) and (6.7) via propagating the η to each of the K hyperplanes. Such a simplification allows for a direct application of the Langrange multiplers to derive the dual. We also will remove the slack variables from the derivations to simplify our expressions, as the relaxations to our objective that we present below makes these slacks redundant. With these simplifications, we rewrite our modified P_2 as:

$$\begin{aligned} \min_{\mathbf{W}\in\mathscr{S}_K,\mathbf{b}} & \frac{1}{2}\left\|\mathbf{W}_1-\mathbf{W}_2\right\|_F^2+\frac{1}{2}\left\|\mathbf{b}_1-\mathbf{b}_2\right\|^2 \\ \text{subject to} & \eta-\left(\mathbf{W}_{1j}^T\mathbf{x}_i+\mathbf{b}_1\right)\leq 0, \quad \forall j\in[K], i\in[n] \\ & \eta+\left(\mathbf{W}_{2j}^T\mathbf{x}_i+\mathbf{b}_2\right)\leq 0, \quad \forall j\in[K], i\in[n]. \end{aligned} \tag{6.14}$$

Using the fact that $\mathbf{W}^\top\mathbf{W}=\mathbf{I}_K$, and using non-negative dual variables $\mathbf{Y},\mathbf{Z}\in\mathbb{R}_+^{K\times n}$, we have the following Lagrangian formulation of (6.14):

$$\begin{aligned} L(\mathbf{W},\mathbf{b},\mathbf{Y},\mathbf{Z}) := & -\operatorname{Tr}\mathbf{W}_1^\top\mathbf{W}_2+\frac{1}{2}\left\|\mathbf{b}_1-\mathbf{b}_2\right\|^2+ \\ & \operatorname{Tr}\mathbf{Y}^\top\left(\eta\mathbf{1}_{K\times n}-\mathbf{W}_1^\top\mathbf{X}-\mathbf{b}_1\mathbf{1}_n^\top\right)+ \\ & \operatorname{Tr}\mathbf{Z}^\top\left(\mathbf{W}_2^\top\mathbf{X}+\mathbf{b}_2\mathbf{1}_n^\top+\eta\mathbf{1}_{K\times n}\right). \end{aligned} \tag{6.15}$$

A straightforward reduction provides the following dual:

$$\begin{aligned} \min_{\mathbf{Y},\mathbf{Z}\in\mathbb{R}_+^{K\times n}} & \frac{1}{2}\mathbf{1}_n^\top\mathbf{Y}^\top\mathbf{Y}\mathbf{1}_n+\operatorname{Tr}\mathbf{Y}\mathbb{K}\mathbf{Z}^T-\eta\operatorname{Tr}(\mathbf{Y}+\mathbf{Z})^\top\mathbf{1}_{K\times n} \\ & \text{s. t. } (\mathbf{Y}-\mathbf{Z})\mathbf{1}_n=\mathbf{0} \qquad (6.16) \\ & \mathbf{Y}\mathbb{K}\mathbf{Y}^\top=\mathbf{Z}\mathbb{K}\mathbf{Z}^\top=\mathbf{I}_K, \qquad (6.17) \end{aligned}$$

where $\mathbb{K}=\mathbf{X}^\top\mathbf{X}$ is a linear kernel, however could be replaced by any other positive definite kernel via the kernel trick. We call our formulation above as *kernelized one-class discriminative subspaces* (KODS). Recall that, for $\mathbb{K}\in\mathbb{R}^{n\times n}\succ 0$, the constraints in (6.17) pose the KODS objective on the generalized Stiefel manifold $\mathscr{G}_{\mathbb{K}}^{K\times n}$, formally defined as $\mathscr{G}_{\mathbb{K}}^{K\times n}=\left\{\Lambda\in\mathbb{R}^{K\times n}:\Lambda\mathbb{K}\Lambda^\top=\mathbf{I}_K,\mathbb{K}\succ 0\right\}$. However, there are two constraints in our ob-

[2]We attempted to use other GODS variants, however they resulted in objectives that seemed computational expensive.

jective that adds hurdle to directly using this manifold for optimization: (i) the null-space constraint in (6.16), and (ii) the requirement that the dual variables are non-negative. Below, we present soft-constraints circumventing these challenges and derive an approximate KODS objective.

Approximate KODS Formulation

We avoid the null-space constraint in KODS via incorporating (6.16) as a soft-constraint into the KODS objective using a regularization penalty, $\lambda > 0$. To circumvent the non-negative constraints, we replace the dual variables $\mathbf{Y}, \mathbf{Z}$ by their element-wise squares, e.g., $\mathbf{Y} \in \mathbb{R}^{K\times n} : \mathbf{Y} \to (\mathbf{Y} \odot \mathbf{Y})$, while retaining $\mathbf{Y} \in \mathscr{G}_{\mathbb{K}}^{K\times n}$. Note that the latter heuristic has been used before, such as in approximating quadratic assignment problems [214]. With these changes, we provide our *approximate KODS formulation* as:

$$\begin{aligned}\min_{\mathbf{Y},\mathbf{Z}\in\mathscr{G}_{\mathbb{K}}^{K\times n}} \mathscr{K}(\mathbf{Y},\mathbf{Z}) = \frac{1}{2}\mathbf{1}_n^\top (\mathbf{Y}\odot\mathbf{Y})^\top (\mathbf{Y}\odot\mathbf{Y})\,\mathbf{1}_n \\ + \operatorname{Tr}(\mathbf{Y}\odot\mathbf{Y})\,\mathbb{K}\,(\mathbf{Z}\odot\mathbf{Z})^T \\ - \eta \operatorname{Tr}((\mathbf{Y}\odot\mathbf{Y}) + (\mathbf{Z}\odot\mathbf{Z}))^\top \mathbf{1}_{K\times n} \\ + \frac{\lambda}{2}\|((\mathbf{Y}\odot\mathbf{Y}) - (\mathbf{Z}\odot\mathbf{Z}))\mathbf{1}\|^2, \end{aligned} \tag{6.18}$$

where the last factor corresponds to (6.16). Note that we use the squared form only on the optimization variables, and not on the constraints, and thus our objective is still on the generalized Stiefel product manifold.

To derive the classification rules at test time (in the next section), we will need expressions for the primal variables in terms of the duals, which we provide below:

$$\mathbf{W}_1(.) = (\mathbf{Z}\odot\mathbf{Z})\,\mathbb{K}(\mathbf{X},.) \tag{6.19}$$

$$\mathbf{W}_2(.) = -(\mathbf{Y}\odot\mathbf{Y})\,\mathbb{K}(\mathbf{X},.) \tag{6.20}$$

$$\mathbf{b}_1 = \operatorname{rowmax}(\eta - (\mathbf{Z}\odot\mathbf{Z})\,\mathbb{K}) \tag{6.21}$$

$$\mathbf{b}_2 = \operatorname{rowmin}(-\eta + (\mathbf{Y}\odot\mathbf{Y})\,\mathbb{K}), \tag{6.22}$$

where rowmax and rowmin corresponds to the maximum and minimum values along the rows of the respective matrices.

6.3 Inference

During inference, we use the decision functions with the learned parameters to classify a given data point as in-class or out-of-class. Specifically, for a new data point $\mathbf{x}$, it is classified as in-class if the following criteria is met:

$$\min(\mathbf{W}_1(\mathbf{x})+\mathbf{b}_1) \geq \eta \wedge \max(\mathbf{W}_2(\mathbf{x})+\mathbf{b}_2) \leq -\eta, \tag{6.23}$$

where the variables $\mathbf{W}$ and $\mathbf{b}$ are either learned in the KODS formalism or the GODS. In case, we have access to a validation set consisting of in-class and out-of-class data (for which we know the class labels), then we may calibrate the threshold η to improve our decision rules. Specifically, suppose we have access to m such validation data points, denoted $\mathscr{X}_v$. Then, to estimate an updated threshold η', we propose to compute the decision scores $v_l = \{\min(\mathbf{W}_1(\mathbf{x})+\mathbf{b}_1)\}_{\mathbf{x}\in\mathscr{X}_v}$ and $v_u = \{\max(\mathbf{W}_2(\mathbf{x})+\mathbf{b}_2)\}_{\mathbf{x}\in\mathscr{X}_v}$. Next, we apply K-Means (or spectral clustering) on v_l and v_u with $K=2$ clusters. Suppose c_{lk} and c_{uk} $(k=1,2)$ are the respective centroids for the two clustering problems; then we propose to update η' as the average of the smaller of the two centroids thresholded by η; i.e.,

$$\Delta\eta = \frac{1}{2}\left([\eta - \min(c_{l1},c_{l2})]_+ - [\eta + \min(c_{u1},c_{u2})]_+\right), \tag{6.24}$$

and use $\eta' = \eta + \Delta\eta$ to form the new decision rules in (6.23).

6.4 GODS Optimization

In contrast to OC-SVM and SVDD, the GODS formulation in (6.10) is non-convex due to the orthogonality constraints on $\mathbf{W}_1$ and $\mathbf{W}_2$.[3] However, these constraints naturally impose a geometry on the solution space and in our case, puts optimization on the Stiefel manifold [134] – a Riemannian manifold characterizing the space of all orthogonal frames. There exist several schemes for geometric optimization over Riemannian manifolds (see [3] for a detailed survey) from which we use the Riemannian conjugate gradient (RCG) scheme in this chapter, due to its stable and fast convergence. In the following, we review some essential components of the RCG scheme and provide the necessary formulae for using it to solve our objectives.

[3] Note that the function $\max(0,\min(z))$ for z in some convex set is also non-convex.

6.4.1 Riemannian Conjugate Gradient

Recall that the standard (Euclidean) conjugate gradient (CG) method [3][Sec.8.3] is a variant of the steepest descent method, however chooses its descent along directions conjugate to previous descent directions with respect to the parameters of the objective. Formally, suppose $F(\mathbf{W})$ represents our objective.[4] Then, the CG method uses the following recurrence at the k-th iteration:

$$\mathbf{W}^k = \mathbf{W}^{k-1} + \lambda^{k-1}\alpha^{k-1}, \tag{6.25}$$

where λ is a suitable step-size (found using line-search) and $\alpha^{k-1} = -\operatorname{grad} F(\mathbf{W}^{k-1}) + \mu^{k-1}\alpha^{k-2}$, where $\operatorname{grad} F(\mathbf{W}^{k-1})$ defines the gradient of F at $\mathbf{W}^{k-1}$ and α^{k-1} is a direction built over the current residual, which is conjugate to previous descent directions (see [3][pp.182])).

When $\mathbf{W}$ belongs to a curved Riemannian manifold, we may use the same recurrence, however there are a few important differences from the Euclidean CG case, namely (i) we need to ensure that the updated point $\mathbf{W}^k$ belongs to the manifold, (ii) there exists efficient vector transports[5] for computing α^{k-1}, and (iii) the gradient grad is along tangent spaces to the manifold. For (i) and (ii), we may resort to computationally efficient retractions (using QR factorizations; see [3][Ex.4.1.2]) and vector transports [3][pp.182], respectively. For (iii), there exist standard ways that take as input a Euclidean gradient of the objective (i.e., assuming no manifold constraints exist), and maps them to the Riemannian gradients [3][Chap.3]. Specifically, for the Stiefel manifold, let $\nabla_{\mathbf{W}} F(\mathbf{W})$ define the Euclidean gradient of F (without the manifold constraints), then the Riemannian gradient is given by:

$$\operatorname{grad} F(\mathbf{W}) = \nabla_{\mathbf{W}} F(\mathbf{W}) - \mathbf{W}\nabla_{\mathbf{W}} F(\mathbf{W})^\top \mathbf{W}. \tag{6.26}$$

The direction $\operatorname{grad} F(\mathbf{W})$ corresponds to a curve along the manifold, descending along which ensures the optimization objective is decreased (atleast locally).

Now, getting back to our one-class objective, all we need to derive to use the RCG, is compute the Euclidean gradients $\nabla_{\mathbf{W}} F(\mathbf{W})$ of our objective in GODS with regard to the variables $\mathbf{W}_j$s. The other variables, such as the biases and slacks, belong to the Euclidean space and their gradients are straightforward. The expression for the Euclidean gradient of

[4]The other optimization variable – $\mathbf{b}$, belongs to the Euclidean manifold, and thus $(\mathbf{W}, \mathbf{b}) \in \mathscr{S}K \times \mathbb{R}^K$. However for brevity and focus, we omit these variables from our optimization discussion.

[5]This is required for computing α_{k-1} that involves the sum of two terms in potentially different tangent spaces, which would need vector transport for moving between them; see [3][pp.182].

our objective with respect to the $\mathbf{W}$'s is given by:

$$\frac{\partial F}{\partial \mathbf{W}_1} = \sum_{i=1}^{n} \mathbf{x}_i \left(\mathbf{W}_1^T \mathbf{x}_i + \mathbf{b}_1\right)^T - \mathbf{Z}_{k_i^*} \left[\eta - \mathbf{W}_{1,k_i^*}^T \mathbf{x}_i - b_1\right]_+, \tag{6.27}$$

$$\frac{\partial F}{\partial \mathbf{W}_2} = \sum_{i=1}^{n} \mathbf{x}_i \left(\mathbf{W}_2^T \mathbf{x}_i + \mathbf{b}_2\right)^T + \mathbf{Z}_{k_i^*} \left[\eta + \mathbf{W}_{2,k_i^*}^T \mathbf{x}_i + b_2\right]_+, \tag{6.28}$$

where $k_i^* \in [K]$ denotes the hyperplane index for the respective subspaces; $k_i^* = \arg\min_k (\mathbf{W}_1^T \mathbf{x}_i + \mathbf{b}_1)$ for (6.27) and $k_i^* = \arg\max_k (\mathbf{W}_2^T \mathbf{x}_i + \mathbf{b}_2)$ for (6.28). The variable $\mathbf{Z}_{k_i^*}$ is a $d \times K$ matrix with all zeros, except k_i^*-th column, which is set to $\mathbf{x}_i$.

6.4.2 KODS Optimization

In this section, we will derive the gradients for our approximate KODS formulation provided in (6.18). Similar to (6.26), the mapping from the Euclidean gradient to the Riemannian gradient for the generalized Stiefel manifold is provided in the following theorem.

Theorem 2. *For the optimization problem* $\min_{\mathbf{U}} \mathscr{K}(\mathbf{U})$ *s.t.* $\mathbf{U}\mathbb{K}\mathbf{U}^\top = \mathbf{I}_K$, $\mathbb{K} \succ 0$, *if* $\nabla_{\mathbf{U}} \mathscr{K}(\mathbf{U})$ *denotes the Euclidean gradient of* $\mathscr{K}(\mathbf{U})$, *then the Riemannian gradient under the canonical metric is given by:*

$$\operatorname{grad} \mathscr{K}(\mathbf{U}) = \nabla_{\mathbf{U}} \mathscr{K}(\mathbf{U}) \mathbb{K}^{-1} - \mathbf{U} \nabla_{\mathbf{U}} \mathscr{K}(\mathbf{U})^\top \mathbf{U}. \tag{6.29}$$

Proof. The proof follows directly from the results in [55][Section 4.5]. □

For the retraction of the iterates on to the manifold, we use the generalized polar decomposition [88] as suggested in [17]. As in the previous section, next we derive the expressions for the Euclidean gradients.

Proposition 1. *Let* $f(\mathbf{Y})$ *be a differentiable matrix function, then the matrix gradient* $\nabla_{\mathbf{Y}^\top} f(\mathbf{Y}) = (\nabla_{\mathbf{Y}} f(\mathbf{Y}))^\top$.

Lemma 1. *For matrices* $\mathbf{Y}, \mathbf{A}$, *and* $\mathbf{D}$ *of appropriate sizes, if* $f(\mathbf{Y}) = \operatorname{Tr}(\mathbf{Y} \odot \mathbf{A})\mathbf{D}$, *then* $\nabla_{\mathbf{Y}} f(\mathbf{Y}) = \mathbf{A} \odot \mathbf{D}^T$.

Proof. If $\mathbf{a}_{i:}, \mathbf{y}_{i:}, \mathbf{d}_{:j}$ represent the i-th row and j-th column of matrices $\mathbf{A}, \mathbf{Y}, \mathbf{D}$ respectively, then

$$f(\mathbf{Y}) = \sum_i (\mathbf{a}_{i:} \odot \mathbf{y}_{i:})^\top \mathbf{d}_{:i} = \sum_i \mathbf{y}_{i:}^\top (\mathbf{a}_{i:} \odot \mathbf{d}_{:i}). \tag{6.30}$$

Then, the gradient w.r.t. $\mathbf{y}_{ij}$, i.e., $\nabla_{\mathbf{y}_{ij}} f(\mathbf{Y}) = \mathbf{a}_{ij} \odot \mathbf{d}_{ji}$, and we have the desired result. □

Lemma 2. *Let* $f(\mathbf{Y}) = \mathrm{Tr}\,(\mathbf{Y}\odot\mathbf{Y})^\top(\mathbf{Y}\odot\mathbf{Y})\mathbf{D}$*, where* $\mathbf{D}$ *is a symmetric matrix. Then,* $\nabla_{\mathbf{Y}} f(\mathbf{Y}) = 4\mathbf{Y}\odot(\mathbf{Y}\odot\mathbf{Y})\mathbf{D}$.

Proof. To simplify the notation, let us use $\mathbf{A} = (\mathbf{Y}\odot\mathbf{Y})$, then using Proposition 1, and applying chain-rule:

$$\begin{aligned}\nabla_{\mathbf{Y}} f(\mathbf{Y}) &= 2\left(\nabla_{\bullet^\top}\mathrm{Tr}\,(\bullet^\top\odot\mathbf{Y}^\top)\mathbf{A}\mathbf{D}\right)^\top \\ &\qquad + 2\nabla_{\bullet}\mathrm{Tr}\mathbf{A}^\top(\bullet\odot\mathbf{Y})\mathbf{D} \\ &= 2\left(\mathbf{Y}^\top\odot(\mathbf{A}\mathbf{D})^\top\right)^\top + 2\mathbf{Y}\odot\left(\mathbf{D}\mathbf{A}^\top\right)^\top \qquad (6.31)\\ &= 2\mathbf{Y}\odot(\mathbf{A}\mathbf{D}) + 2\mathbf{Y}\odot(\mathbf{A}\mathbf{D}^\top),\end{aligned}$$

where we used Lemma 1 to obtain (6.31). Using the symmetry of $\mathbf{D}$, we have the result. □

Theorem 3. *Let* $\mathbf{E}_n = \mathbf{1}_n\mathbf{1}_n^\top$*,* $\mathbf{Y}^2 = \mathbf{Y}\odot\mathbf{Y}$*, and* $\mathbf{Z}^2 = \mathbf{Z}\odot\mathbf{Z}$*, the Euclidean gradient of* $\mathscr{K}(\mathbf{Y},\mathbf{Z})$ *in* (6.18) *is:*

$$\begin{aligned}\nabla_{\mathbf{Z}}\mathscr{K}(\mathbf{Y},\mathbf{Z}) &= 2\lambda\mathbf{Z}\odot\mathbf{Z}^2\mathbf{E}_n + \mathbf{Z}\odot\mathbf{Y}^2[2\mathbb{K} - \lambda\mathbf{E}_n] - 2\eta\mathbf{Z}\\ \nabla_{\mathbf{Y}}\mathscr{K}(\mathbf{Y},\mathbf{Z}) &= \gamma\mathbf{Y}\odot\mathbf{Y}^2\mathbf{E}_n + \mathbf{Y}\odot\mathbf{Z}^2[2\mathbb{K} - \lambda\mathbf{E}_n] - 2\eta\mathbf{Y},\end{aligned}$$

where $\gamma = 2 + 2\lambda$.

Proof. The result directly follows by applying Lemma 1 and Lemma 2 to the formulation in (6.18). □

6.4.3 Optimization Initialization

Due to the non-convexity of our objective, there could be multiple local solutions. To this end, we resort to the following initialization of our optimization variables, which we found to be empirically beneficial. Specifically, for the GODS optimization, we first sort all the training points based on their Euclidean distances from the origin. Next, we randomly select a suitable number ($3K$ in our experiments) of such sorted points near and far from the origin, compute a compact singular value decomposition (thin-SVD) of these points, and initialize the GODS subspaces using these orthonormal frames from the SVD. The intercepts ($\mathbf{b}$) are initialized to zero. For KODS, we initialize the dual variables as $\frac{1}{nK}$.

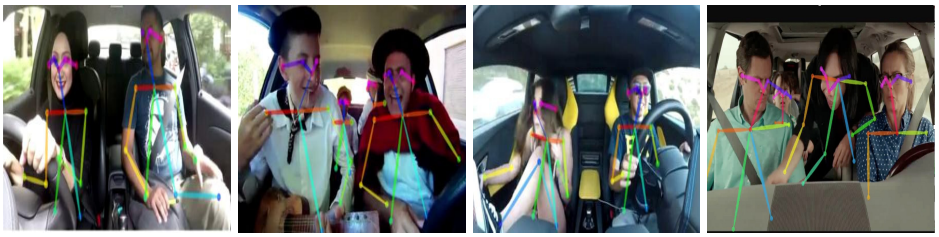


Fig. 6.3 Frames from our Dash-Cam-Pose dataset. The left-top frame has poses in-position (one-class), while the rest of the frames are from videos labeled out-of-position.

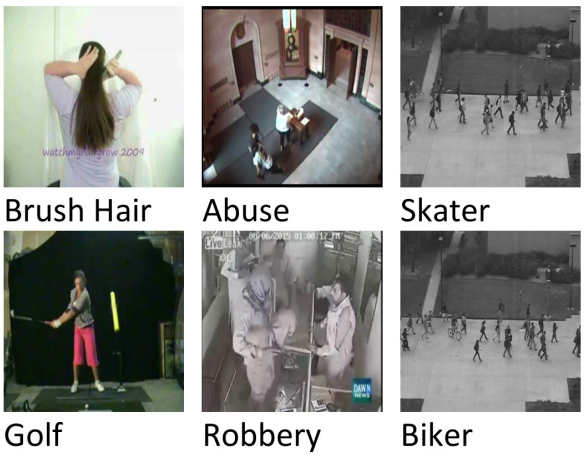


Fig. 6.4 Some examples from JHMDB (first column), UCF-Crime (second column) and USCD Ped2 (third column) datasets, with respective categories.

6.5 Experiments

In this section, we provide experiments demonstrating the performance of our proposed schemes on several one-class tasks. We will introduce these tasks and the associated datasets briefly next along with detailing the data features used.

6.5.1 Dash-Cam-Pose Dataset

Out-of-position (OOP) human pose detection is an important problem with regard to the safety of passengers in a vehicle. While, there are public datasets for human pose estimation, they are usually annotated for generic pose estimation tasks, and neither do they contain any in-vehicle poses as captured by a dashboard camera, nor are they annotated for pose anomalies. To this end, we collected 104 videos, each 20-30 min long, from the Internet (including Youtube, ShutterStock, and Hollywood road movies). As these videos were originally recorded for diverse reasons, there are significant shifts in camera angles, perspectives, locations of the camera, scene changes, etc. We encourage the interested reader to refer

to [203] for more details on this dataset and its collection. Next, we manually selected clips from these videos that are found interesting for our task. To extract as many clips as possible from these videos, we first segmented each video into three second clips at 30fps, which resulted in approximately 7000 clips. Next, we selected only those clips where the camera is approximately placed on the dashboard looking inwards, which amounted to 4,875 clips, totalling 4.06 hours. We annotated each clip with a weak binary label based on the poses of humans in the front seat. Specifically, if all the front-seat humans (passengers and the driver) are seated in-position, the clip was given a positive label, while if any human is seated OOP (based on [137, 54]) for the entire 3s, the clip was labeled as negative. Next, we used Open Pose [21] on each clip to extract a sequence of poses for every person. Our final Dash-Cam-Pose dataset consists of 4875 short videos, 1.06 million poses, of which 310,996 are OOP.

We explore two pose-sequence representations for this task: (i) a simple bag-of-words (BoW) model, and (ii) using a Temporal Convolutional Network (TCN) [104] consisting of residual units with 1D convolutional layers capturing both local and global information via convolutions for each joint across time. For BoW, we use 1024 pose centroids computed using K-Means clustering. For the TCN, we use the following procedure. The poses from each person in each frame are vectorized and stacked into the temporal dimension. For each pose thus passed through TCN, we extract features from the last pooling layer, using a model pre-trained on the NTU-RGBD dataset [165] (for 3D skeleton action recognition) to produce 256-D features for every clip. As our pre-trained TCN model takes 3D poses as input, we pad our Dash-Cam-Poses with zeros in the third dimension.

We use a four-fold cross-validation for evaluation on Dash-Cam-Pose. Specifically, we divide the entire dataset into four non-overlapping splits, each split consisting of approximately 1/4-th the dataset, of which roughly 2/3rd's are labeled as positive (in-pose) and the rest as OOP. We use only the positive data in each split to train our one-class models. Once the models are trained, we evaluate on the held out split. For every embedded-pose feature, we use the binary classification accuracy against the ground truth. The evaluation is repeated on all the four splits and the performance averaged.

6.5.2 Data Preprocessing

JHMDB dataset: To evaluate the performance on the entire dataset, we cycle over the 21 classes, and the scores are averaged. For representing the frames, we use an ImageNet pre-trained VGG-16 model and extract features from the 'fc-6' layer (4096-D).

UCF-Crime dataset: To encode the videos, we use the state-of-the-art Inflated-3D (I3D) neural network [24]. Specifically, video frames from non-overlapping sliding windows (8

frames each) are passed through the I3D network; features are extracted from the 'Mix_5c' network layer, that are then reshaped to 2048-D vectors. For anomaly detections on the test set, we first map back the features classified as anomalies by our scheme to the frame-level and apply the official evaluation metrics [180].

UCSD Ped2 dataset: To encode the video data, we apply the deep autoencoder with causal 3D convolutions [7] trained to minimize the reconstruction loss. We extract features from the bottleneck layer of this model to be input to our algorithm. As the videos can be of arbitrary length, the pipeline is trained on clips from temporal sliding windows with a stride of one and consisting of 16 frames. As anomalous events are labeled frame-wise in this dataset, we use the averaged clip-level predictions within a window as the prediction for the center frame in that window. We use the evaluation metrics on these frame-level predictions similar to [123, 127, 1].

6.5.3 Experimental Setup

Before using the above features in our algorithms, we found that it is beneficial to unit-normalize them. However, we do report results without such normalization on other datasets in Section 6.5.7. These scaled features are then used in our GODS formulations, the optimization schemes for which are implemented using ManOpt [17] and PyManOpt [187]. We use the conjugate gradient scheme for optimization, which typically converges in about 200 iterations. We initialize the iterates using the approach described in Section 6.4.3. The hyper-parameters in our models are chosen via cross-validation, and the sensitivities of these parameters are evaluated in the next section. We use regularization constants $\nu = 1$ and $\lambda = 1$. We use the inference criteria described in Section 6.3 for classifying a test point as in-class or an anomaly.

6.5.4 Evaluation Metrics

On the UCF-Crime dataset, we follow the official evaluation protocol, reporting AUC as well as the false alarm rate. For other datasets, we use the $F1$ score to reflect the sensitivity and accuracy of our classification models. As the datasets we use - especially the Dash-Cam-Pose – are imbalanced across the two classes, having a single performance metric over the entire dataset may fail to characterize the quality of the discrimination for each class separately, which is of primary importance for the one-class task. To this end, we also report True Negative Rate $TNR = \frac{TN}{N}$, Negative Predictive Value $NPV = \frac{TN}{TN+FN}$, and $\overline{F1} = \frac{2 \times TNR \times NPV}{TNR+NPV}$, alongside standard F1 scores. We will use $\overline{F1}$ on the Dash-Cam-Pose

dataset and $F1$ score on other datasets. Informally, $\overline{F1}$ is the same as $F1$ with the positive and negative categories switched.

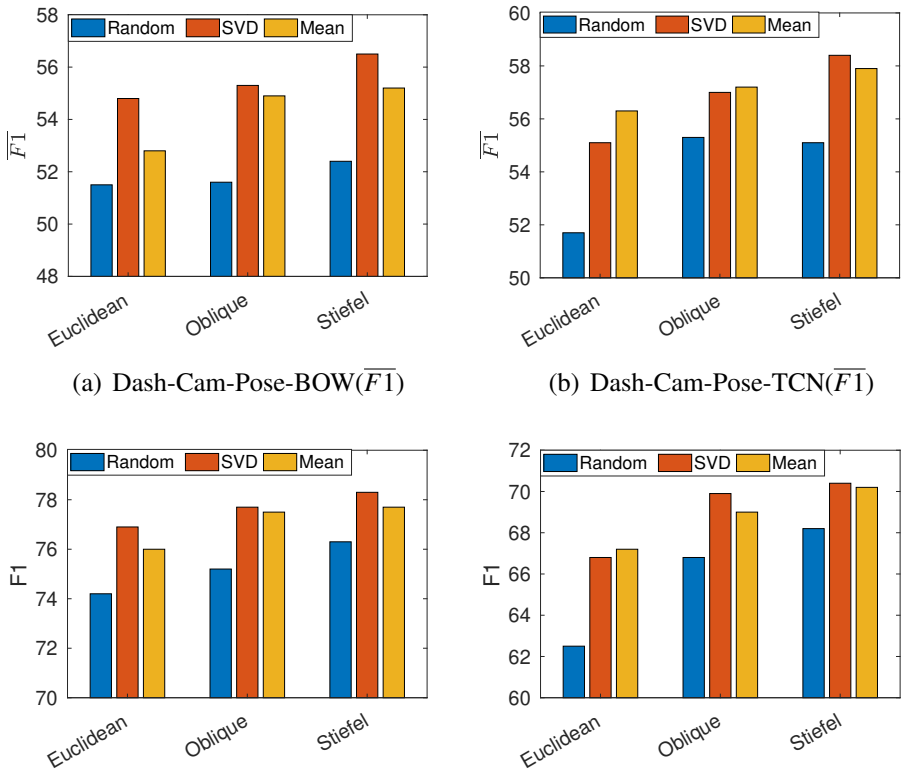
6.5.5 Ablative Studies

Synthetic Experiments: To gain insights into the inner workings of our schemes, we present results on several 2D synthetic toy datasets. In Figure 6.1(a)–6.1(c), we show three plots with 100 points distributed as (i) Gaussian and (ii) some arbitrary distribution[6]. We show the BODS hyperplanes in the Figure 6.1(a), and the GODS 2D subspaces in Figures 6.1(b), 6.1(c) with the hyperplanes belonging to each subspace shown in same color. As the plots show, our models are able to orient the subspaces such that they confine the data within a minimal volume. In Figure 6.1(d), we show 300 data points (black dots) distributed along a 2D ring, a situation when a rectilinear GODS may fail (as the inner circle does not contain the one-class). As seen in Figures 6.1(e) and 6.1(f), the two kernelized KODS hyperplanes capture the outer and inner decision regions separately, and their combined decision region is able to capture the ring structure of the input data, as seen in Figure 6.1(d). In Figure 6.1(g), we plot the decision surfaces for 3D data points.

Choice of Manifold and Initialization. As described in Section 6.2.3, our GODS algorithm may assume several optimization manifolds based on the type of regularization used between the classifier hyperplanes. In the Figure 6.5, we evaluate three different manifold choices, namely (i) Stiefel manifold, (ii) oblique manifold, and (iii) Euclidean manifold. We also evaluate three different hyperplane initialization strategies: (i) random, (ii) SVD (as describved in Section 6.4.3), and (iii) mean of the data features. From the Figure, it can be seen that the Stiefel manifold and SVD initialization works best compared to oblique and Euclidean manifolds and against random or mean initializations, consistently on the three datasets.

Sensitivity of Margin η. The hyperparameter η decides the support margin between the hyperplanes and the one-class data. In Figure 6.6, we analyze the performance sensitivity against changes in η on the JHMDB and the Dash-Cam-Pose datasets. To ensure the learning will not ignore the changes in η, we increased the regularization constants on the two terms involving η in (6.4). On both datasets, the TPR (true positive rate) increases for increasing η, while the TNR decreases with a higher value of η. This is because the distances between the two orthonormal frames $(\mathbf{W}_1, \mathbf{W}_2)$ may become larger to satisfy the new margin constraints imposed by η. Thus, more points will be included between the two frames and classified as positive. As $F1$ relies on the classification sensitivity of the positive data, while the $\overline{F1}$ relies

[6]The data follows the formula $f(x) = \sqrt{x} * (x + sign(randn) * rand)$, where randn and rand are standard MATLAB functions.



(a) Dash-Cam-Pose-BOW($\overline{F1}$) (b) Dash-Cam-Pose-TCN($\overline{F1}$)

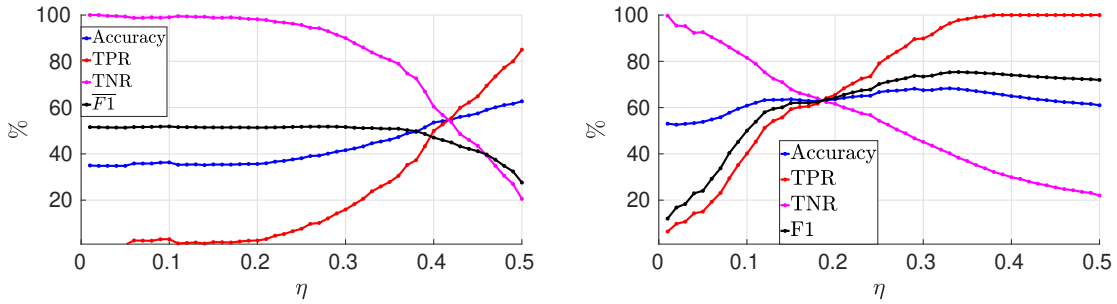
(c) JHMDB($F1$) (d) UCF-Crime($F1$)

Fig. 6.5 Performance of GODS with $F1$ and $\overline{F1}$ for different initialization methods and manifold assumptions.

on the negative classifications, they show opposite trends. Observing these trends, we fix $\eta = 0.3$ in our subsequent experiments.

Number of Hyperplanes K. In Figure 6.7, we plot the influence of increasing number of hyperplanes on our four datasets. We find that after a certain number of hyperplanes, the performance saturates, which is expected, and suggests that more hyperplanes might lead to overfitting to the positive class. We also find that the TCN embedding is significantly better than the BoW model (by nearly 3%) on the Dash-Cam-Pose dataset when using our proposed methods. Surprisingly, S-SVDD is found to perform quite inferior against ours; note that this scheme learns a low-dimensional subspace to project the data to (as in PCA), and applies SVDD on this subspace. We believe, these subspaces perhaps are common to the negative points as well that it cannot be suitably discriminated, leading to poor performance. We make a similar observation on the other datasets as well.

Kernel Choices and Number of Subspaces in K. In Figure 6.8, we demonstrate the performance of KODS on the datasets for various choices of the embedding kernels, and also



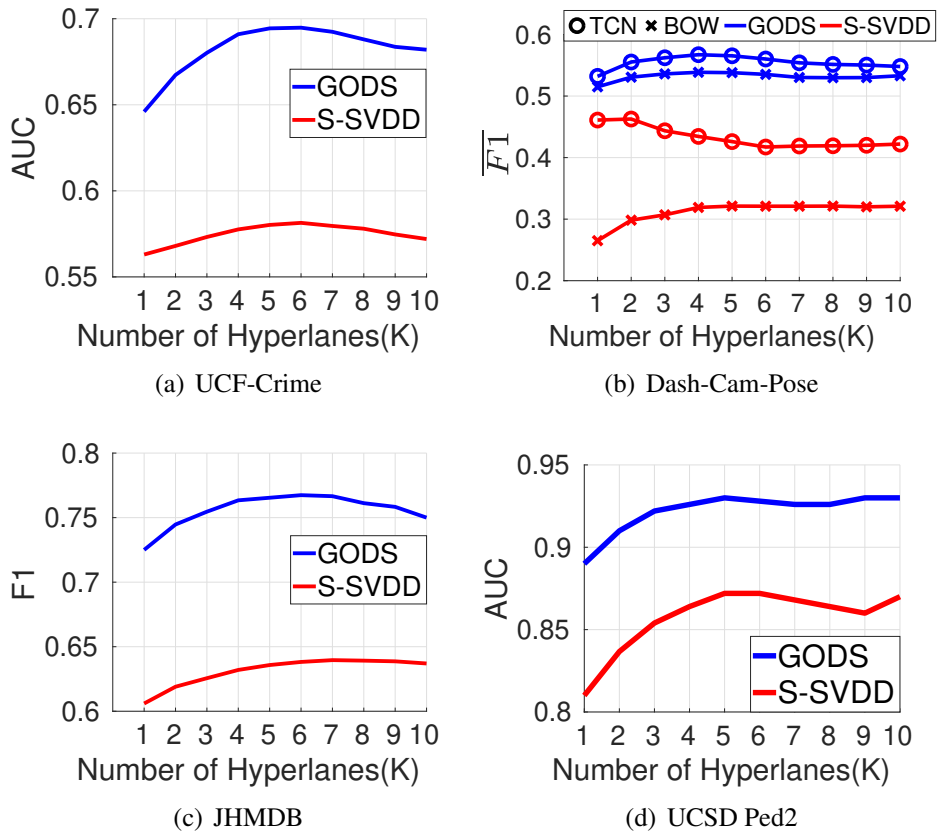
(a) Dash-Cam-Pose($\overline{F1}$) (b) JHMDB($F1$)

Fig. 6.6 Performance of BODS with $F1$ and $\overline{F1}$ for increasing η.

when increasing the number of Hilbert space classifiers K for each choice of the kernel feature map on every dataset we use. Specifically, we experiment with linear, RBF, and polynomial kernels on JHMDB, UCF-Crime, UCSD Ped2, and Dash-Cam-Pose (with TCN) datasets, while use the Chi-square [132] and Histogram Intersection kernels [8] on the Dash-Cam-Pose dataset with the Bag-of-Words features. We set $\sigma = 0.1$ for the bandwidth in the RBF kernel, polynomial kernel degree is set to 3, and use 1024 words in the Bag-of-Words representation. In Figure 6.8, we see that the performance saturates with increasing number of hyperplanes. The RBF kernel seems to work better on the JHMDB and UCF-Crime datasets, while the polynomial kernel demonstrates higher performances on the UCSD-Ped2 and the Dash-Cam-Pose datasets (with TCN). For the Bag-of-Words features on the Dash-Cam-Pose dataset, the Chi-square kernel shows better performance than the Historgram Intersection kernel.

Empirical Convergence. In Figures 6.9(a) and 6.9(b), we show the empirical convergences of our GODS algorithm on the JHMDB dataset using the original GODS formulation in (6.10). We show the convergence in the objective value as well as the magnitude of the Riemannian gradients. As is clear, our algorithm convergences in about 200 iterations on this dataset. We repeat this experiment on the KODS formulation (6.18) using different kernel maps. As is seen from Figures 6.9(c) and 6.9(d), while the convergence is slower compared to that in GODS – perhaps due to our approximations – it does converge suitably for appropriate kernel choices.

Running Time. In the Figure 6.9(e), we demonstrate the time taken for training our different models. For this analysis, we use an Intel i7-6800K 3.4GHz CPU with 6 cores. We implement the different algorithms in the Matlab, run it on the same data, and record the training time with an increasing number of training samples. For GODS, KODS, and S-SVDD, we use 3 hyperplanes in the subspaces. It can be seen that the GODS, BODS, and KODS algorithms



(a) UCF-Crime (b) Dash-Cam-Pose

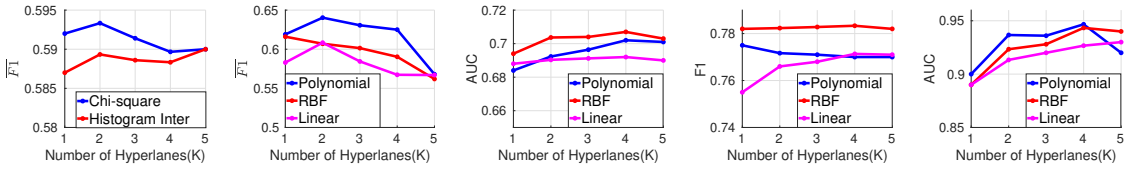
(c) JHMDB (d) UCSD Ped2

Fig. 6.7 Performance of GODS for an increasing number of subspaces.

are not substantially more computationally expensive in comparison to other prior methods, while remaining empirically superior (Table 6.1).

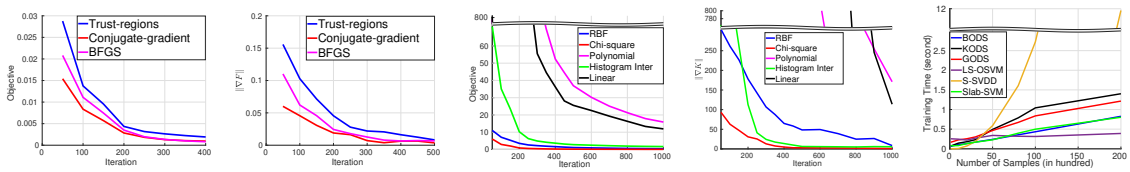
6.5.6 State-of-the-Art Comparisons

In Table 6.1, we compare our variants to the state-of-the-art methods. As alluded to earlier, for our Dash-Cam-Pose dataset, as its positive and negative classes are imbalanced, we resort to reporting the $\overline{F1}$ score on the test set. From the table, our variants are seen to outperform prior methods by a significant margin; especially our GODS and KODS schemes demonstrate the best performances on different tasks. For example, using TCN, KODS outperforms other kernelized prior variants by over 20%. Similarly, on the JHMDB dataset, both GODS and KODS are better than the next best kernel-based method (K-OC-SVM) by about 20%, and improves the classification accuracy by over 30%. Overall, the experiments clearly substantiate the performance benefits afforded by our methods on the one-class task.



(a) Dash-Cam-Pose-BOW (b) Dash-Cam-Pose-TCN (c) UCF-Crime (d) JHMDB (e) UCSD Ped2

Fig. 6.8 Performance of KODS in different kernel type on various datasets for an increasing number of subspaces.



(a) GODS-objective (b) GODS-gradient (c) KODS-objective (d) KODS-gradient (e) Running time

Fig. 6.9 (a–d) show optimization convergence on JHMDB dataset. (a-b) GODS (6.10) against the choice of optimization schemes, (c-d) KODS (6.18) under different kernels (using CG optimizer). (e) compares running time.

In Table 6.2-left, we present results against the state of the art on the UCF-Crime dataset using the AUC metric and false alarm rates; we use the standard threshold of 50%. While, our results are lower than [180] by 4% in AUC and 0.2 larger in false alarm rate, their problem setup is completely different from ours in that they use weakly labeled abnormal videos as well in their training, which we do not use and which as per definition is not a one-class problem. Thus, our results are incomparable to theirs. Against other methods on this dataset, our schemes are about 5-10% better. In the Table 6.2-right, we provide the performance (AUC) on the UCSD Ped2 dataset. Compared with the recent state-of-the-art methods, both GODS and KODS achieves similar performances using the 3D autoencoder features. Among these methods, Luo et al. [126] and Liu et al. [127] propose ConvLSTM-AE and S-RNN respectively, which rely on the recurrent neural networks, that might be hard to train. Abati et al. [1] also uses the 3D autoencoder features similar to ours, but also employs additional constraints for building the Conditional Probability Density (CPD), which is more expensive compared to our solution. We also note that Table 6.2-right lists prior methods that use deep learning models, such as the ConvLSTM autoencoder [126], Stacked-RNN [127], and GANs [149]; our GODS variants offer competitive performances against them.

Table 6.1 Average performances on the Dash-Cam-Pose and JHMDB datasets. Dash-Cam-Pose uses the $\overline{F1}$ score while JHMDB uses $F1$ score as evaluation metric (classification accuracy is shown in the brackets). K-OC-SVM and K-SVDD are the RBF kernelized variants.

Method	CarPose_BOW	CarPose_TCN	JHMDB
OC-SVM [164]	0.167 (0.517)	0.279(0.527)	0.301 (0.568)
SVDD [184]	0.448 (0.489)	0.477(0.482)	0.407 (0.566)
K-OC-SVM [164]	0.327 (0.495)	0.361(0.491)	0.562 (0.412)
K-SVDD [184]	0.476 (0.477)	0.489 (0.505)	0.209 (0.441)
K-PCA [92]	0.145 (0.502)	0.258 (0.492)	0.245 (0.557)
Slab-SVM [67]	0.468 (0.568)	0.498 (0.577)	0.643 (0.637)
KNFST [14]	0.345 (0.487)	0.368 (0.496)	0.667 (0.501)
KNN [81]	0.232 (0.475)	0.276 (0.488)	0.643 (0.492)
LS-OSVM [40]	0.234 (0.440)	0.246(0.460)	0.663(0.582)
S-SVDD [176]	0.325 (0.490)	0.464 (0.500)	0.642 (0.498)
BODS	0.523 (0.582)	0.532 (0.579)	0.725 (0.714)
GODS	0.553 (0.629)	0.584 (0.601)	0.777 (0.752)
KODS	**0.596 (0.642)**	**0.664 (0.604)**	**0.785 (0.726)**

Table 6.2 Performances on UCF-Crime dataset (left) and UCSD Ped2 dataset (right). *Setup is different.

UCF Crime Dataset			**UCSD Ped2 Dataset**	
Method	**AUC**	**FAR**	**Method**	**AUC**
Random	0.50	-	ConvLSTM-AE. [126]	0.88
Hasan et al. [80]	0.51	27.2	GANs [149]	0.88
Lu et al. [124]	0.66	3.1	S-RNN [127]	0.92
*Waqas et al. [180]	0.75	1.9	Autoregression [1]	**0.95**
Sohrab et al. [176]	0.59	10.5	FFP+MC, Liu et al. [123]	**0.95**
BODS	0.68	2.7	BODS	0.91
GODS	0.70	2.1	GODS	0.93
KODS	**0.71**	**2.1**	KODS	**0.95**

Table 6.3 Performances on UCI datasets. N is number of samples, D is the feature dimension and T is the target (positive) class. Experiments 1 to 7 are OC-SVM, K-OC-SVM, SVDD, S-SVDD, K-SVDD, GODS and KODS. The result is in percentage.

Dataset	N	D	T	1	2	3	4	5	6	7
Sonar	208	60	Mines	53.5	56.5	62.5	63.8	60.9	71.6	72.5
Pump	1500	64	Normal	63.2	60.1	84.6	85.7	83.6	87.6	88.2
Scale	625	4	Left	68.8	67.6	70.3	90.7	73.4	92.3	92.5
Survival	306	3	Survived	64.4	74.3	83.4	84.1	83.5	87.6	88.1
Banknote	1372	5	No	65.7	70.0	76.4	90.8	80.4	94.7	95.8

Table 6.4 Comparisons of GODS variants on UCI datasets. We compare: (i) *GODS* (6.10) using the Stiefel manifold, (ii) GODS_N using the non-compact Stiefel (6.11), (iii) the Euclidean (6.13), and (iv) the oblique manifolds (6.12). We compare under (i) ℓ_2 unit-normalization of inputs and (ii) C, under soft-orthogonality ((6.12), (6.13)). We report F1 scores (in %) and standard deviations over 5 trials.

	Type	Sonar	Pump	Scale	Survival	Banknote
GODS	ℓ_2	**71.6±2.6**	**87.6±0.6**	89.2±5.4	87.6±1.9	94.7±3.8
	$\neq\ell_2$	69.2±3.9	85.5±0.9	92.3±4.5	86.5±8.9	90.5±2.1
	$\neq$	68.3±4.1	85.2±0.8	92.2±4.1	87.0±9.5	90.4±2.1
	ℓ_2 + C					
	ℓ_2 + C	71.6±3.7	87.0±0.6	88.7±3.6	**87.8±1.8**	95.1±1.7
GODS_N	ℓ_2	69.2±8.6	86.7±7.1	84.8±7.0	85.3±0.3	90.9±6.2
	$\neq\ell_2$	68.7±1.5	85.9±5.1	89.7±9.1	82.9±4.8	82.4±2.6
	$\neq$	66.9±2.2	85.7±4.1	89.5±9.5	85.2±7.2	88.4±4.5
	ℓ_2 + C					
	ℓ_2 + C	69.1±5.1	86.4±0.6	86.9±4.2	86.3±7.8	90.9±6.4
GODS_E	ℓ_2	66.7±2.3	85.9±0.3	79.8±5.1	84.8±1.7	93.8±6.7
	$\neq\ell_2$	68.1±4.5	85.8±0.6	94.4±6.5	85.9±1.1	90.6±3.0
	$\neq$	68.2±3.0	85.9±0.6	95.0±2.7	86.5±7.5	91.4±2.2
	ℓ_2 + C					
	ℓ_2 + C	68.7±2.4	86.2±0.2	81.0±3.9	86.5±1.2	94.1±1.5
GODS_θ	ℓ_2	69.1±3.5	87.0±0.6	81.6±3.7	85.7±6.1	94.0±1.2
	$\neq\ell_2$	69.3±3.5	84.6±1.2	97.1±3.6	85.5±1.5	94.1±3.9
	$\neq$	70.5±4.6	86.5±0.7	**97.1±2.9**	85.9±8.7	95.3±3.2
	ℓ_2 + C					
	ℓ_2 + C	70.6±4.0	87.3±0.7	82.3±4.3	85.9±3.3	**95.6±3.1**

6.5.7 Performance on UCI datasets

As the reader might acknowledge, the algorithms proposed in this chapter are not specialized to only computer vision datasets, but could be applied for the anomaly detection task on any data mining, machine learning, or robotics task. To this end, in Table 6.3, we evaluate GODS and KODS on five datasets downloaded from UCI datasets[7] and TU delft pattern recognition lab website[8]. These datasets are (i) sonar, the task in which is to discriminate between sonar signals bounced off a metal cylinder and those bounced off a roughly cylindrical rock, (ii) the Delft pump dataset, the task in which is to detect abnormal condition of a submersible pump, (iii) the Scale dataset is to classify the balance scale tip to the right, tip to the left, or be balanced, (iv) the Haberman's survival dataset records the survival of patients who had undergone surgery for breast cancer, and (v) the Banknote dataset is to detect if the feature from an image passing the evaluation of an authentication procedure for banknotes.

We follow the evaluation protocol in the recent paper [176] for all the datasets and compare the performances to those reported in that paper. Specifically, the evaluation uses a split of 70/30 for the positive class; the model training is performed on the 70%, and tested on the remaining 30% positive class data and the negative (anomalous) data (which is not used in training). We repeat the split in the positive class five times and report the average performance on the five trials. For datasets having more than two classes, we pick one class as positive, while the remaining as negative, and follow the same protocol as above. In Table 6.3, we report the performances of GODS and KODS against those reported in [176]. In the table, N represents the number of samples in the dataset, D denotes the feature dimension of each sample, and T is the target class picked as positive (same as in [176]). We use three hyperplanes in the GODS subspaces, and set the sensitivity margin $\eta = 0.3$ in both GODS and KODS. For KODS, we use polynomial kernel with degree as 3. As is clear from the table, we outperform the previous state-of-the-art results on all datasets. Specifically, GODS is substantially better than the previous best method S-SVDD by 2–8%, and the KODS is even better by 1–2%. This is because our GODS algorithm better characterizes the data distribution from positive classes and thus producing higher cost for anomalies during the inference. For KODS, the kernel embedding would further bring advantages in learning the decision regions better fitting the normal samples.

In the Table 6.4, we evaluate the various extensions of GODS as described in Section 6.2.3. Through these experiments, we evaluate the impact of unit-normalization on the data inputs and the orthogonality assumptions on the hyperplanes, and analyze the adequacy of each variant when such assumptions may not be relevant. In the Table 6.4, each column contains

[7] http://archive.ics.uci.edu/ml/index.php

[8] http://homepage.tudelft.nl/n9d04/occ/index.html

the results for one dataset while the four rows in one column are the result for one variant of GODS. From top to the bottom, we have 4 settings relaxing different constraints; they are: 1) unit normalization applied on the inputs, denoted ℓ_2, 2) no unit norm is enforced, $\neq \ell_2$, 3) $\neq \ell_2$ but with soft-orthogonality constraints (*C*) as described in (6.12), (6.13), and 4) ℓ_2 norm and *C* are used together. We also report the standard deviations associated with each experiment over the five trials.

From the experimental results, it is found that the orthogonal constraint is generally helpful, whenever applicable. For instance, the results in the third and fourth row in Euclidean and Oblique manifolds are better than the ones without orthogonal constraints by up to 2%. In terms of the ℓ_2 norm constraints, it depends on the nature of the data points. For example, in the Scale dataset, each dimension of the data captures the presence of some semantic attribute and thus ℓ_2 norm may not make much sense on them. However, for the vision datasets or the other four UCI datasets, the feature normalization could bound the data allowing the one-class model to better capture the distribution.

6.6 Chapter Summary

In this Chapter, we presented a novel one-class video representation learning formulation, which we called GODS, using pairs of complementary classifiers; these classifiers are oriented in such a way as to circumscribe the data density within a rectilinear space of minimal volume. We also explored variants of our scheme via relaxing the various assumptions in our problem setup and using kernel feature maps. Due to the orthonormality we impose on the classifiers, our objectives are non-convex, and solving for which we resorted to Riemannian optimization frameworks on the Stiefel manifold and its variants. We presented experiments on a diverse set of anomaly detection tasks, demonstrating state-of-the-art performances. We further analyzed the generalizability of our framework to non-vision data by presenting experiments on five UCI datasets; our results outperforming prior baselines by significant margins. More recent works do not follow the idea of this paper, building max margin optimization problem for anomaly detection. They have progressed in a different direction of relying on the deep neural network [57, 223, 186] due to the efficiency gain of neural network in the large-scaled dataset. Although these work also achieve promising result, our work shares the similar core idea with them that is to maximise the separability between normal and abnormal samples.

Chapter 7

Conclusion and Future Work

In this thesis, we worked on the topic of discriminative video representation learning. To address this problem, we propose several video representation learning algorithms achieving the state-of-the-art performances in various video-based applications, such as action recognition, action detection and one-class anomaly detection. The proposed method can be treated as unsupervised/weakly-supervised smart pooling function that can be applied on both shallow hand-crafted features and deep features from convolutional neural networks. In this concluding chapter, we will summarize the major contribution of this thesis and discuss alternative paths to take in the future work.

7.1 Contributions

Capturing the motion dynamics, especially the long-term dynamics, is a critical step in the video representation learning. To model longer temporal information, previous works tend to trim the entire video sequence into short clips and apply global average pooling in the last or intermediate layer. However, this strategy 1) fails to learn the temporal relations across clips, 2) equally treat noise and background frames and clips, that hurt the representation learning. This thesis proposed several novel pooling schemes as alternatives in both deep and non deep learning scenarios, involving a wider scope of video representation learning including the concept of rank pooling, contrastive learning, universal adversarial perturbation, one-class video representation learning.

In Chapter 3, we present dynamic flow images, a novel temporal ordered video representation that summarizes a set of consecutive optical flow frames in a video sequence as a single two-channel image. This early fusion scheme is based on a Ranking SVM [64] formulation that minimizes a quadratic objective with temporal order constraints. After solving this formulation, the parameters learnt encapsulate the temporal ordering of the pixels from frame

to frame, effectively describing the underlying action dynamics. By aggregating the whole video sequence into one or a few image-like inputs, this technique dramatically reduces the computing cost as compared to prior methods. Moreover, using optical flow images reduces the amount of noise/background pixels that are presented in RGB images. At last, we presented an unique three-stream CNN architecture that combines single RGB frames (for action context), stack of flow pictures (for local action dynamics), and our innovative dynamic flow stream as an extra CNN input cue (for long range action evolution), achieving the state-of-the-art performance in multiple benchmarks.

In Chapter 4, we offer an unique SVM pooling strategy with temporally-ordered constraint for learning more discriminative video representation without assigning equal priority to all frames as in Chapter 3. Specifically, we utilise a multiple instance learning framework to solve a binary SVM classification problem and learn meaningful decision boundary on video features against background/noise characteristics. This decision boundary serves as a video representation that collects and summarises the discriminative characteristics in a video sequence while also explicitly capturing the action dynamics. To improve efficiency, we investigate several versions of our optimisation problem and offer increasingly less expensive inference techniques, such as a joint pooling and classification objective, as well as an end-to-end learnable CNN architecture. We evaluate the proposed SVM descriptor over eight challenging benchmarks by using both hand-crafted features and CNN features. The findings show that our technique regularly outperforms state-of-the-art methods in terms of video representation quality.

In Chapter 5, we follow the idea and motivation in Chapter 4, proposing discriminative pooling scheme for video representation learning. Instead of using additional data or synthetic data as negatives, we introduce adversarial perturbations into the set-up, casting the problem as weekly supervised contrastive video representation learning. Given a well-trained CNN model, we learn universal adversarial perturbations for building the negative. And then, we formulate a binary classification problem to learn temporally-ordered discriminative subspaces that separate the data features from their perturbed counterparts. Because perturbation encapsulates the most sensitive dimension of features, such subspaces must fit to relevant sections of the features in order to achieve excellent performance, and therefore produce discriminative and robust video representation. Furthermore, we present effective Riemannian optimisation techniques for achieving our goal on the Stiefel manifold. Experiments on a variety of datasets investigated the efficacy of each component in our approach, demonstrating best-in-class performance on benchmarks.

In Chapter 6, we extend video representation schemes from previous Chapters into a new topic of one-class video representation learning, that only provide positive class label while

require detecting both positives and negatives during inference. Specifically, we reformulate the previous pooling formulation into a new one-class learning formulation that employs pairs of complimentary classifiers. These classifiers are arranged in such a manner that the data density is circumscribed inside a rectilinear region with a minimum volume. Our objectives are non-convex due to the orthogonality we impose on the classifiers, and we used Riemannian optimisation frameworks on the Stiefel manifold and its variations to solve them. We also looked at several versions of our method by loosening up some of the assumptions in our issue setting and employing kernel feature maps. Experiments on data from many computer vision applications, such as anomaly detection in video sequences, human postures, and human activities, show the empirical benefits of our method. By presenting tests on five UCI datasets, we investigated the generalizability of our approach to non-vision data; our results outperformed earlier baselines by a substantial margin.

7.2 Ethical and Societal Impacts Review

All datasets in this thesis are publicly available without including features or label information about individual names, gender, race, sexuality, or other protected characteristics. In addition, we has received the consent to use or share all data used in this thesis for non-commercial purpose. We implement the adversarial attacks for generating adversarial perturbation in the Chapter 5. However, this is only for learning better video representation in the following max-margin problem, which will not have any potential negative security considerations.One major negative societal impacts of deep learning is that the massive training would generate huge carbon emissions which is harm for the environment. While this thesis also mainly adopts deep learning framework, our method proposed discriminative pooling algorithm in the early and middle layers to alleviate the video redundancy issue which significantly improve the efficiency and reduce the computational cost. Moreover, our method proposed in the Chapter 6 is a non deep learning setup, which achieves promising result with cheaper cost compared to other deep learning works. one potential negative societal impacts of this thesis is that the method in the Chapter 6 is able to analyse bulk surveillance data and predict anomaly behaviour from the surveillance data, which can be used for criminal profiling.

7.3 Future Works

Formulation with Universal Adversarial Perturbation

In Chapter 5, we introduce a discriminative pooling scheme for video representation learning, which learn a set of hyperplanes, as a subspace, to distinguish the encoded frame-

level CNN features from their perturbed counterparts created by adversarial perturbations. Then, The decision subspace is used as the video descriptor. To solve this problem, we adopt block-coordinate optimization scheme, which fixes the CNN layers and learn universal adversarial perturbation first and then solves the perturbation to update the CNN model.

An interesting direction is to generate a joint formulation to update the universal adversarial perturbation and CNN model at the same time, which can be treated as a self-calibration system for supervised deep learning with additional supervisory signal figuring out which dimension of the CNN feature is less robust. Moreover, this scheme can be applied in most of the computer vision applications.

One-class Video Representation Learning

An interesting direction to extend Chapter 6 is perhaps to use more than two classifiers in the GODS framework. While, we experimented with a variant of this idea using multiple orthonormal frames within our set-up, we found that it did not showcase any empirical benefits. We presume this inferior performance is perhaps due to the lack of appropriate regularizations across the classifiers and definite complementarity conditions between the classifiers and the data.

Further, there are several aspects of our scheme that needs rigorous treatment. For example, deriving generalization bounds on GODS is one such. Analysis of the representation complexity within a computation learning theory framework is yet another direction. An analysis of our optimization landscape is a direction that could help better initialize our schemes. Extending our framework as a module within an end-to-end neural network is an interesting direction as well.

Bibliography

[1] Abati, D., Porrello, A., Calderara, S., and Cucchiara, R. (2019). Latent space autoregression for novelty detection. In *CVPR*, pages 481–490. (cited on pages: 88, 106, 111, and 112)

[2] Absil, P.-A. and Gallivan, K. A. (2006). Joint diagonalization on the oblique manifold for independent component analysis. In *ICASSP*. IEEE. (cited on page: 97)

[3] Absil, P.-A., Mahony, R., and Sepulchre, R. (2009). *Optimization algorithms on matrix manifolds*. Princeton University Press. (cited on pages: 76, 89, 96, 100, and 101)

[4] Arnab, A., Dehghani, M., Heigold, G., Sun, C., Lučić, M., and Schmid, C. (2021). Vivit: A video vision transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6836–6846. (cited on pages: 20, 21, and 68)

[5] Baccouche, M., Mamalet, F., Wolf, C., Garcia, C., and Baskurt, A. (2011). Sequential deep learning for human action recognition. In *Human Behavior Understanding*, pages 29–39. (cited on pages: 4, 19)

[6] Bachman, P., Hjelm, R. D., and Buchwalter, W. (2019). Learning representations by maximizing mutual information across views. In *Advances in Neural Information Processing Systems*, pages 15509–15519. (cited on pages: 69, 71)

[7] Bai, S., Kolter, J. Z., and Koltun, V. (2018). An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*. (cited on page: 106)

[8] Barla, A., Odone, F., and Verri, A. (2003). Histogram intersection kernel for image classification. In *ICIP*, volume 3, pages III–513. IEEE. (cited on page: 109)

[9] Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828. (cited on page: 1)

[10] Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166. (cited on page: 18)

[11] Bertasius, G., Wang, H., and Torresani, L. (2021). Is space-time attention all you need for video understanding? In Meila, M. and Zhang, T., editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 813–824. PMLR. (cited on pages: 20, 68)

[12] Bilen, H., Fernando, B., Gavves, E., and Vedaldi, A. (2017). Action recognition with dynamic image networks. *PAMI*. (cited on pages: 32, 38, and 75)

[13] Bilen, H., Fernando, B., Gavves, E., Vedaldi, A., and Gould, S. (2016). Dynamic image networks for action recognition. In *CVPR*. (cited on pages: 4, 22, 25, 27, 30, 33, 34, 37, 38, 52, 58, 61, 62, 75, and 85)

[14] Bodesheim, P., Freytag, A., Rodner, E., Kemmler, M., and Denzler, J. (2013). Kernel null space methods for novelty detection. In *CVPR*, pages 3374–3381. (cited on page: 112)

[15] Boothby, W. M. (1986). *An introduction to differentiable manifolds and Riemannian geometry*, volume 120. Academic press. (cited on pages: 70, 76, and 89)

[16] Boumal, N., Mishra, B., Absil, P.-A., and Sepulchre, R. (2014a). Manopt, a Matlab toolbox for optimization on manifolds. *Journal of Machine Learning Research*, 15:1455–1459. (cited on page: 52)

[17] Boumal, N., Mishra, B., Absil, P.-A., and Sepulchre, R. (2014b). Manopt, a matlab toolbox for optimization on manifolds. *JMLR*, 15(1):1455–1459. (cited on pages: 80, 102, and 106)

[18] Bunescu, R. C. and Mooney, R. J. (2007). Multiple instance learning for sparse positive bags. In *ICML*. (cited on page: 48)

[19] Caba Heilbron, F., Escorcia, V., Ghanem, B., and Carlos Niebles, J. (2015). Activitynet: A large-scale video benchmark for human activity understanding. In *CVPR*. (cited on page: 55)

[20] Calderara, S., Heinemann, U., Prati, A., Cucchiara, R., and Tishby, N. (2011). Detecting anomalies in people's trajectories using spectral graph analysis. *CVIU*, 115(8):1099–1111. (cited on page: 91)

[21] Cao, Z., Simon, T., Wei, S.-E., and Sheikh, Y. (2016). Realtime multi-person 2d pose estimation using part affinity fields. *arXiv preprint arXiv:1611.08050*. (cited on page: 105)

[22] Carreira, J., Noland, E., Banki-Horvath, A., Hillier, C., and Zisserman, A. (2018). A short note about kinetics-600. *CoRR*, abs/1808.01340. (cited on pages: 14, 64, and 67)

[23] Carreira, J., Noland, E., Hillier, C., and Zisserman, A. (2019). A short note on the kinetics-700 human action dataset. *arXiv preprint arXiv:1907.06987*. (cited on page: 14)

[24] Carreira, J. and Zisserman, A. (2017). Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*. (cited on pages: 5, 14, 19, 20, 54, 64, 66, 67, 72, 83, 84, 85, and 105)

[25] Chalapathy, R., Menon, A. K., and Chawla, S. (2018). Anomaly detection using one-class neural networks. *arXiv preprint arXiv:1802.06360*. (cited on pages: 88, 92)

[26] Chandala, V., Banerjee, A., and Kumar, V. (2009). Anomaly detection: A survey, ACM computing surveys. *University of Minnesota*. (cited on pages: 89, 90)

[27] Chang, X., Yu, Y.-L., Yang, Y., and Xing, E. P. (2017). Semantic pooling for complex event analysis in untrimmed videos. *PAMI*, 39(8):1617–1632. (cited on page: 72)

[28] Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. (2020). A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*. (cited on pages: 22, 69, 71, and 73)

[29] Chen, X. and He, K. (2021). Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15750–15758. (cited on pages: 72, 86)

[30] Chen, X., Xie, S., and He, K. (2021). An empirical study of training self-supervised vision transformers. *arXiv preprint arXiv:2104.02057*. (cited on page: 72)

[31] Cherian, A., Fernando, B., Harandi, M., and Gould, S. (2017a). Generalized rank pooling for activity recognition. In *CVPR*. (cited on pages: 22, 27, 52, 70, 72, 75, 76, 84, and 85)

[32] Cherian, A. and Gould, S. (2018). Second-order temporal pooling for action recognition. *IJCV*. (cited on page: 21)

[33] Cherian, A. and Gould, S. (2019). Second-order temporal pooling for action recognition. *IJCV*, 127(4):340–362. (cited on page: 67)

[34] Cherian, A., Koniusz, P., and Gould, S. (2017b). Higher-order pooling of cnn features via kernel linearization for action recognition. In *WACV*. (cited on page: 21)

[35] Cherian, A., Sra, S., Gould, S., and Hartley, R. (2018). Non-linear temporal subspace representations for activity recognition. In *CVPR*. (cited on pages: 22, 82)

[36] Cherian, A. and Wang, J. (2021). Generalized one-class learning using pairs of complementary classifiers. *IEEE transactions on pattern analysis and machine intelligence*. (cited on page: 7)

[37] Chiang, A. C. (1984). Fundamental methods of mathematical economics. (cited on page: 79)

[38] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*. (cited on page: 18)

[39] Choi, M. J., Torralba, A., and Willsky, A. S. (2012). Context models and out-of-context objects. *Pattern Recognition Letters*, 33(7):853–862. (cited on page: 91)

[40] Choi, Y.-S. (2009). Least squares one-class support vector machine. *Pattern Recognition Letters*, 30(13):1236–1240. (cited on pages: 87, 112)

[41] Chong, Y. S. and Tay, Y. H. (2017). Abnormal event detection in videos using spatiotemporal autoencoder. In *International Symposium on Neural Networks*, pages 189–196. Springer. (cited on page: 92)

[42] Cinbis, R. G., Verbeek, J., and Schmid, C. (2017). Weakly supervised object localization with multi-fold multiple instance learning. *PAMI*, 39(1):189–203. (cited on page: 45)

[43] Cong, Y., Yuan, J., and Liu, J. (2011). Sparse reconstruction cost for abnormal event detection. In *CVPR*, pages 3449–3456. IEEE. (cited on pages: 91, 92)

[44] Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3):273–297. (cited on page: 43)

[45] Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *ECCV*. (cited on page: 91)

[46] Dalal, N., Triggs, B., and Schmid, C. (2006). Human detection using oriented histograms of flow and appearance. In *ECCV*, pages 428–441. Springer. (cited on page: 91)

[47] Del Giorno, A., Bagnell, J. A., and Hebert, M. (2016). A discriminative framework for anomaly detection in large videos. In *ECCV*. (cited on page: 90)

[48] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. In *CVPR*, pages 248–255. IEEE. (cited on page: 92)

[49] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*. (cited on page: 20)

[50] Donahue, J., Anne Hendricks, L., Guadarrama, S., Rohrbach, M., Venugopalan, S., Saenko, K., and Darrell, T. (2015). Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*. (cited on page: 19)

[51] Dontchev, A. L. and Rockafellar, R. T. (2009). Implicit functions and solution mappings. *Springer Monogr. Math.* (cited on page: 53)

[52] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*. (cited on pages: 20, 68)

[53] Du, Y., Wang, W., and Wang, L. (2015). Hierarchical recurrent neural network for skeleton based action recognition. In *CVPR*. (cited on pages: 4, 19)

[54] Duma, S. M., Kress, T. A., Porta, D. J., Woods, C. D., Snider, J. N., Fuller, P. M., and Simmons, R. J. (1996). Airbag-induced eye injuries: a report of 25 cases. *Journal of Trauma and Acute Care Surgery*, 41(1):114–119. (cited on page: 105)

[55] Edelman, A., Arias, T. A., and Smith, S. T. (1998). The geometry of algorithms with orthogonality constraints. *SIAM journal on Matrix Analysis and Applications*, 20(2):303–353. (cited on pages: 89, 102)

[56] Elgendy, M. (2020). *Deep Learning for Vision Systems*. Manning Publications. (cited on page: 2)

[57] Fang, Z., Liang, J., Zhou, J. T., Xiao, Y., and Yang, F. (2020). Anomaly detection with bidirectional consistency in videos. *IEEE Transactions on Neural Networks and Learning Systems*. (cited on page: 115)

[58] Faugeras, O. (1993). *Three-dimensional computer vision: a geometric viewpoint*. MIT press. (cited on page: 79)

[59] Feichtenhofer, C., Pinz, A., and Wildes, R. (2016a). Spatiotemporal residual networks for video action recognition. In *NIPS*. (cited on pages: 61, 67, and 84)

[60] Feichtenhofer, C., Pinz, A., and Wildes, R. P. (2017a). Spatiotemporal multiplier networks for video action recognition. In *CVPR*. (cited on pages: 14, 15, 16, 58, 67, and 84)

[61] Feichtenhofer, C., Pinz, A., and Wildes, R. P. (2017b). Temporal residual networks for dynamic scene recognition. In *CVPR*. (cited on pages: 14, 15, 16, 24, 42, 54, 66, 67, 81, and 84)

[62] Feichtenhofer, C., Pinz, A., and Zisserman, A. (2016b). Convolutional two-stream network fusion for video action recognition. In *CVPR*. (cited on pages: 4, 32, 34, 37, 38, 58, and 61)

[63] Fernando, B., Anderson, P., Hutter, M., and Gould, S. (2016). Discriminative hierarchical rank pooling for activity recognition. In *CVPR*. (cited on pages: 27, 37, and 67)

[64] Fernando, B., Gavves, E., Oramas, J. M., Ghodrati, A., and Tuytelaars, T. (2015). Modeling video evolution for action recognition. In *CVPR*. (cited on pages: 22, 25, 26, 52, 58, 62, 67, 75, 85, and 117)

[65] Fernando, B. and Gould, S. (2016). Learning end-to-end video classification with rank-pooling. In *ICML*. (cited on page: 27)

[66] Fernando, B. and Gould, S. (2017). Discriminatively learned hierarchical rank pooling networks. *International Journal of Computer Vision*. (cited on pages: 27, 38)

[67] Fragoso, V., Scheirer, W., Hespanha, J., and Turk, M. (2016). One-class slab support vector machine. In *ICPR*, pages 420–425. IEEE. (cited on page: 112)

[68] Gardner, A. B., Krieger, A. M., Vachtsevanos, G., and Litt, B. (2006). One-class novelty detection for seizure analysis from intracranial eeg. *JMLR*, 7(Jun):1025–1044. (cited on page: 90)

[69] Gärtner, T., Flach, P. A., Kowalczyk, A., and Smola, A. J. (2002). Multi-instance kernels. In *ICML*. (cited on pages: 46, 48)

[70] Girdhar, R., Ramanan, D., Gupta, A., Sivic, J., and Russell, B. (2017a). ActionVLAD: Learning spatio-temporal aggregation for action classification. In *CVPR*. (cited on page: 21)

[71] Girdhar, R., Ramanan, D., Gupta, A., Sivic, J., and Russell, B. (2017b). Actionvlad: Learning spatio-temporal aggregation for action classification. In *CVPR*. (cited on page: 67)

[72] Girshick, R. (2015). Fast R-cnn. In *ICCV*. (cited on page: 92)

[73] Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial networks. *arXiv preprint arXiv:1406.2661*. (cited on page: 11)

[74] Gould, S., Fernando, B., Cherian, A., Anderson, P., Cruz, R. S., and Guo, E. (2016). On differentiating parameterized argmin and argmax problems with application to bi-level optimization. *arXiv preprint arXiv:1607.05447*. (cited on pages: 53, 79)

[75] Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P. H., Buchatskaya, E., Doersch, C., Pires, B. A., Guo, Z. D., Azar, M. G., et al. (2020). Bootstrap your own latent: A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733*. (cited on pages: 22, 71, 72, and 86)

[76] Hadsell, R., Chopra, S., and LeCun, Y. (2006). Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742. IEEE. (cited on pages: 69, 71)

[77] Hager, W. W. and Zhang, H. (2005). A new conjugate gradient method with guaranteed descent and an efficient line search. *SIAM Journal on optimization*, 16(1):170–192. (cited on page: 80)

[78] Harandi, M. T., Salzmann, M., and Hartley, R. (2014a). From manifold to manifold: Geometry-aware dimensionality reduction for SPD matrices. In *ECCV*. (cited on page: 70)

[79] Harandi, M. T., Salzmann, M., Jayasumana, S., Hartley, R., and Li, H. (2014b). Expanding the family of grassmannian kernels: An embedding perspective. In *ECCV*. (cited on page: 76)

[80] Hasan, M., Choi, J., Neumann, J., Roy-Chowdhury, A. K., and Davis, L. S. (2016). Learning temporal regularity in video sequences. In *CVPR*, pages 733–742. (cited on pages: 88, 92, and 112)

[81] Hautamaki, V., Karkkainen, I., and Franti, P. (2004). Outlier detection using k-nearest neighbour graph. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, volume 3, pages 430–433. IEEE. (cited on page: 112)

[82] Hayat, M., Bennamoun, M., and An, S. (2015). Deep reconstruction models for image set classification. *PAMI*, 37(4):713–727. (cited on pages: 14, 15, 54, 65, and 67)

[83] Hayat, M., Khan, S. H., and Bennamoun, M. (2017). Empowering simple binary classifiers for image set based face recognition. *IJCV*, pages 1–20. (cited on page: 67)

[84] He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. (2019). Momentum contrast for unsupervised visual representation learning. *arXiv preprint arXiv:1911.05722*. (cited on pages: 22, 69, 71, 73, and 86)

[85] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer. (cited on page: 11)

[86] Heller, K. A., Svore, K. M., Keromytis, A. D., and Stolfo, S. J. (2003). One class support vector machines for detecting anomalous windows registry accesses. In *Workshop on Data Mining for Computer Security*. (cited on page: 90)

[87] Hénaff, O. J., Razavi, A., Doersch, C., Eslami, S., and Oord, A. v. d. (2019). Data-efficient image recognition with contrastive predictive coding. *arXiv preprint arXiv:1905.09272*. (cited on pages: 69, 71)

[88] Higham, N. J., Mehl, C., and Tisseur, F. (2010). The canonical generalized polar decomposition. *SIAM Journal on Matrix Analysis and Applications*, 31(4):2163–2180. (cited on page: 102)

[89] Hjelm, R. D., Fedorov, A., Lavoie-Marchildon, S., Grewal, K., Bachman, P., Trischler, A., and Bengio, Y. (2018). Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv:1808.06670*. (cited on pages: 69, 71)

[90] Hochreiter, S., Bengio, Y., Frasconi, P., Schmidhuber, J., et al. (2001). Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. (cited on page: 18)

[91] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780. (cited on page: 18)

[92] Hoffmann, H. (2007). Kernel PCA for novelty detection. *Pattern recognition*, 40(3):863–874. (cited on page: 112)

[93] Huang, Z., Wang, R., Shan, S., and Chen, X. (2015). Projection metric learning on grassmann manifold with application to video based face recognition. In *CVPR*. (cited on page: 70)

[94] Itti, L. and Koch, C. (2000). A saliency-based search mechanism for overt and covert shifts of visual attention. *Vision research*, 40(10-12):1489–1506. (cited on page: 91)

[95] Jegou, H., Perronnin, F., Douze, M., Sánchez, J., Perez, P., and Schmid, C. (2012). Aggregating local image descriptors into compact codes. *TPAMI*, 34(9):1704–1716. (cited on page: 14)

[96] Jhuang, H., Gall, J., Zuffi, S., Schmid, C., and Black, M. J. (2013). Towards understanding action recognition. In *ICCV*. (cited on pages: 24, 90)

[97] Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., and Darrell, T. (2014). Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678. ACM. (cited on pages: 32, 61)

[98] Joachims, T. (2002). Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142. ACM. (cited on page: 29)

[99] Judd, T., Ehinger, K., Durand, F., and Torralba, A. (2009). Learning to predict where humans look. In *ICCV*. (cited on page: 91)

[100] Kar, A., Rai, N., Sikka, K., and Sharma, G. (2017). Adascan: Adaptive scan pooling in deep convolutional neural networks for human action recognition in videos. In *CVPR*. (cited on page: 67)

[101] Kay, W., Carreira, J., Simonyan, K., Zhang, B., Hillier, C., Vijayanarasimhan, S., Viola, F., Green, T., Back, T., Natsev, P., et al. (2017). The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*. (cited on pages: 23, 42, and 54)

[102] Khan, S. S. and Madden, M. G. (2009). A survey of recent trends in one class classification. In *Irish conference on artificial intelligence and cognitive science*, pages 188–197. Springer. (cited on page: 90)

[103] Kim, J. and Grauman, K. (2009). Observe locally, infer globally: a space-time mrf for detecting abnormal activities with incremental updates. In *CVPR*. IEEE. (cited on page: 91)

[104] Kim, T. S. and Reiter, A. (2017). Interpretable 3D human action analysis with temporal convolutional networks. (cited on pages: 14, 15, 54, 65, 67, and 105)

[105] Krawczyk, B. (2015). One-class classifier ensemble pruning and weighting with firefly algorithm. *Neurocomputing*, 150:490–500. (cited on page: 90)

[106] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *NIPS*. (cited on pages: 4, 11, 12, 15, 32, and 64)

[107] Kuehne, H., Jhuang, H., Garrote, E., Poggio, T., and Serre, T. (2011). HMDB: a large video database for human motion recognition. In *ICCV*. (cited on pages: 14, 23, 24, 26, 42, 53, and 80)

[108] Kumar, N., Berg, A. C., Belhumeur, P. N., and Nayar, S. K. (2009). Attribute and simile classifiers for face verification. In *ICCV*. (cited on pages: 24, 42, and 54)

[109] Lai, K.-T., Yu, F. X., Chen, M.-S., and Chang, S.-F. (2014). Video event detection by inferring temporal instance labels. In *CVPR*. (cited on page: 46)

[110] Lazarevic, A., Ertoz, L., Kumar, V., Ozgur, A., and Srivastava, J. (2003). A comparative study of anomaly detection schemes in network intrusion detection. In *ICDM*. SIAM. (cited on page: 90)

[111] Lazimy, R. (1982). Mixed-integer quadratic programming. *Mathematical Programming*, 22(1):332–349. (cited on page: 49)

[112] Lee, I., Kim, D., Kang, S., and Lee, S. (2017). Ensemble deep learning for skeleton-based action recognition using temporal sliding LSTM networks. In *ICCV*. (cited on page: 84)

[113] Lee, K., Lee, K., Lee, H., and Shin, J. (2018). A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *NIPS*. (cited on page: 92)

[114] Li, C., Han, Z., Ye, Q., and Jiao, J. (2013a). Visual abnormal behavior detection based on trajectory sparse reconstruction analysis. *Neurocomputing*, 119:94–100. (cited on page: 91)

[115] Li, Q., Qiu, Z., Yao, T., Mei, T., Rui, Y., and Luo, J. (2016). Action recognition by learning deep multi-granular spatio-temporal video representation. In *ICMR*. (cited on pages: 4, 19, and 38)

[116] Li, W., Mahadevan, V., and Vasconcelos, N. (2013b). Anomaly detection and localization in crowded scenes. *TPAMI*, 36(1):18–32. (cited on page: 90)

[117] Li, W. and Vasconcelos, N. (2015). Multiple instance learning for soft bags via top instances. In *CVPR*. (cited on page: 45)

[118] Li, W., Yu, Q., Divakaran, A., and Vasconcelos, N. (2013c). Dynamic pooling for complex event recognition. In *ICCV*. (cited on page: 46)

[119] Li, W., Zhang, Z., and Liu, Z. (2010). Action recognition based on a bag of 3d points. In *CVPRW*. (cited on pages: 23, 42, and 54)

[120] Liang, S., Li, Y., and Srikant, R. (2017). Enhancing the reliability of out-of-distribution image detection in neural networks. *arXiv preprint arXiv:1706.02690*. (cited on page: 92)

[121] Liang, S., Li, Y., and Srikant, R. (2018). Principled detection of out-of-distribution examples in neural networks. In *ICLR*. (cited on page: 92)

[122] Liu, J., Shahroudy, A., Xu, D., Kot, A. C., and Wang, G. (2017). Skeleton-based action recognition using spatio-temporal LSTM network with trust gates. *arXiv preprint arXiv:1706.08276*. (cited on pages: 67, 84)

[123] Liu, W., Luo, W., Lian, D., and Gao, S. (2018). Future frame prediction for anomaly detection–a new baseline. In *CVPR*, pages 6536–6545. (cited on pages: 88, 106, and 112)

[124] Lu, C., Shi, J., and Jia, J. (2013). Abnormal event detection at 150 fps in Matlab. In *CVPR*, pages 2720–2727. (cited on pages: 91, 112)

[125] Lu, J., Issaranon, T., and Forsyth, D. (2017). Safetynet: Detecting and rejecting adversarial examples robustly. In *ICCV*. (cited on pages: 70, 72)

[126] Luo, W., Liu, W., and Gao, S. (2017a). Remembering history with convolutional lstm for anomaly detection. In *ICME*. IEEE. (cited on pages: 92, 111, and 112)

[127] Luo, W., Liu, W., and Gao, S. (2017b). A revisit of sparse coding based anomaly detection in stacked RNN framework. In *ICCV*, pages 341–349. (cited on pages: 88, 92, 106, 111, and 112)

[128] Masi, I., Tran, A., Hassner, T., Leksut, J. T., and Medioni, G. (2016). Do We Really Need to Collect Millions of Faces for Effective Face Recognition? In *ECCV*. (cited on pages: 54, 65)

[129] Matteoli, S., Diani, M., and Theiler, J. (2014). An overview of background modeling for detection of targets and anomalies in hyperspectral remotely sensed imagery. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 7(6):2317–2336. (cited on page: 90)

[130] Miech, A., Zhukov, D., Alayrac, J.-B., Tapaswi, M., Laptev, I., and Sivic, J. (2019). Howto100m: Learning a text-video embedding by watching hundred million narrated video clips. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2630–2640. (cited on page: 14)

[131] Miyato, T., Maeda, S.-i., Koyama, M., and Ishii, S. (2018). Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1979–1993. (cited on page: 72)

[132] Moh'd A Mesleh, A. (2007). Chi square feature extraction based svms arabic language text categorization system. *Journal of Computer Science*, 3(6):430–435. (cited on page: 109)

[133] Moosavi-Dezfooli, S.-M., Fawzi, A., Fawzi, O., and Frossard, P. (2017). Universal adversarial perturbations. (cited on pages: 70, 72, 74, 80, 82, and 86)

[134] Muirhead, R. J. (2009). *Aspects of multivariate statistical theory*, volume 197. John Wiley & Sons. (cited on page: 100)

[135] Neimark, D., Bar, O., Zohar, M., and Asselmann, D. (2021). Video transformer network. *arXiv preprint arXiv:2102.00719*. (cited on page: 20)

[136] Nolan, D. (1991). The excess-mass ellipsoid. *Journal of multivariate analysis*, 39(2):348–371. (cited on page: 91)

[137] Nordhoff, L. S. (2005). *Motor vehicle collision injuries: biomechanics, diagnosis, and management*. Jones & Bartlett Learning. (cited on page: 105)

[138] Nowozin, S., Bakir, G., and Tsuda, K. (2007). Discriminative subsequence mining for action classification. In *ICCV*. (cited on page: 46)

[139] Oh, S. J., Fritz, M., and Schiele, B. (2017). Adversarial image perturbation for privacy protection–a game theory perspective. In *ICCV*. (cited on pages: 70, 72)

[140] Oord, A. v. d., Li, Y., and Vinyals, O. (2018). Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*. (cited on pages: 69, 71)

[141] Park, S., Kim, W., and Lee, K. M. (2012). Abnormal object detection by canonical scene-based contextual model. In *ECCV*. (cited on page: 91)

[142] Parkhi, O. M., Vedaldi, A., and Zisserman, A. (2015). Deep face recognition. (cited on page: 11)

[143] Pascanu, R., Mikolov, T., and Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In *ICML*. (cited on page: 18)

[144] Peng, X., Wang, L., Wang, X., and Qiao, Y. (2016). Bag of visual words and fusion methods for action recognition: Comprehensive study and good practice. *CVIU*. (cited on pages: 14, 38)

[145] Perera, P. and Patel, V. M. (2018). Learning deep features for one-class classification. *arXiv preprint arXiv:1801.05365*. (cited on page: 92)

[146] Perronnin, F., Sánchez, J., and Mensink, T. (2010). Improving the fisher kernel for large-scale image classification. In *ECCV*. (cited on page: 14)

[147] Pimentel, M. A., Clifton, D. A., Clifton, L., and Tarassenko, L. (2014). A review of novelty detection. *Signal Processing*, 99:215–249. (cited on page: 90)

[148] Popoola, O. P. and Wang, K. (2012). Video-based abnormal human behavior recognition—a review. *IEEE Transactions on Systems, Man, and Cybernetics*, 42(6):865–878. (cited on page: 90)

[149] Ravanbakhsh, M., Nabi, M., Mousavi, H., Sangineto, E., and Sebe, N. (2018). Plug-and-play cnn for crowd motion analysis: An application in abnormal event detection. In *Winter Conference on Applications of Computer Vision (WACV)*, pages 1689–1698. IEEE. (cited on pages: 111, 112)

[150] Ravanbakhsh, M., Nabi, M., Sangineto, E., Marcenaro, L., Regazzoni, C., and Sebe, N. (2017). Abnormal event detection in videos using generative adversarial nets. In *ICIP*. IEEE. (cited on page: 92)

[151] Ravanbakhsh, M., Sangineto, E., Nabi, M., and Sebe, N. (2019). Training adversarial discriminators for cross-channel abnormal event detection in crowds. In *WACV*, pages 1896–1904. IEEE. (cited on pages: 88, 92)

[152] Ren, H., Pan, H., Olsen, S. I., and Moeslund, T. B. (2016). A comprehensive study of sparse codes on abnormality detection. *arXiv preprint arXiv:1603.04026*. (cited on page: 92)

[153] Ridnik, T., Ben-Baruch, E., Noy, A., and Zelnik-Manor, L. (2021). Imagenet-21k pretraining for the masses. *arXiv preprint arXiv:2104.10972*. (cited on page: 21)

[154] Rohrbach, M., Amin, S., Andriluka, M., and Schiele, B. (2012). A database for fine grained activity detection of cooking activities. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1194–1201. IEEE. (cited on pages: 23, 54)

[155] Ruff, L., Goernitz, N., Deecke, L., Siddiqui, S. A., Vandermeulen, R., Binder, A., Müller, E., and Kloft, M. (2018). Deep one-class classification. In *ICML*. (cited on page: 92)

[156] Sabokrou, M., Fayyaz, M., Fathy, M., and Klette, R. (2016). Fully convolutional neural network for fast anomaly detection in crowded scenes. *arXiv preprint arXiv:1609.00866*. (cited on page: 92)

[157] Sabokrou, M., Khalooei, M., Fathy, M., and Adeli, E. (2018). Adversarially learned one-class classifier for novelty detection. In *CVPR*, pages 3379–3388. (cited on pages: 88, 90, and 92)

[158] Sadanand, S. and Corso, J. J. (2012). Action bank: A high-level representation of activity in video. In *CVPR*. (cited on page: 14)

[159] Saleh, B., Farhadi, A., and Elgammal, A. (2013). Object-centric anomaly detection by attribute-based reasoning. In *CVPR*, pages 787–794. (cited on page: 90)

[160] Saligrama, V. and Chen, Z. (2012). Video anomaly detection based on local statistical aggregates. In *CVPR*, pages 2112–2119. IEEE. (cited on page: 90)

[161] Satkin, S. and Hebert, M. (2010). Modeling the temporal extent of actions. In *ECCV*. (cited on page: 45)

[162] Schindler, K. and Van Gool, L. (2008). Action snippets: How many frames does human action recognition require? In *CVPR*. (cited on page: 41)

[163] Schlegl, T., Seeböck, P., Waldstein, S. M., Schmidt-Erfurth, U., and Langs, G. (2017). Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In *IPMI*, pages 146–157. Springer. (cited on pages: 88, 92)

[164] Schölkopf, B., Platt, J. C., Shawe-Taylor, J., Smola, A. J., and Williamson, R. C. (2001). Estimating the support of a high-dimensional distribution. *Neural computation*, 13(7):1443–1471. (cited on pages: 87, 91, and 112)

[165] Shahroudy, A., Liu, J., Ng, T.-T., and Wang, G. (2016). NTU RGB+ D: A large scale dataset for 3d human activity analysis. In *CVPR*. (cited on pages: 23, 42, 54, 80, and 105)

[166] Shi, Y., Tian, Y., Wang, Y., Zeng, W., and Huang, T. (2017). Learning long-term dependencies for action recognition with a biologically-inspired deep network. In *ICCV*. (cited on page: 84)

[167] Sigurdsson, G. A., Divvala, S., Farhadi, A., and Gupta, A. (2017). Asynchronous temporal fields for action recognition. In *CVPR*. (cited on pages: 23, 64, 66, and 67)

[168] Sigurdsson, G. A., Varol, G., Wang, X., Farhadi, A., Laptev, I., and Gupta, A. (2016). Hollywood in homes: Crowdsourcing data collection for activity understanding. In *ECCV*. (cited on pages: 23, 42, 54, and 64)

[169] Simonyan, K., Vedaldi, A., and Zisserman, A. (2013). Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*. (cited on page: 67)

[170] Simonyan, K. and Zisserman, A. (2014a). Two-stream convolutional networks for action recognition in videos. In *NIPS*. (cited on pages: 4, 6, 11, 13, 14, 15, 26, 30, 36, 38, and 80)

[171] Simonyan, K. and Zisserman, A. (2014b). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*. (cited on pages: 32, 34, 58, and 72)

[172] Sivic, J. and Zisserman, A. (2003). Video Google: A text retrieval approach to object matching in videos. In *ICCV*, page 1470. (cited on page: 14)

[173] Smith, S. T. (1994). Optimization techniques on riemannian manifolds. *Fields institute communications*, 3(3):113–135. (cited on page: 76)

[174] Smola, A. J. and Schölkopf, B. (1998). *Learning with kernels*. Citeseer. (cited on page: 52)

[175] Smola, A. J. and Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222. (cited on page: 28)

[176] Sohrab, F., Raitoharju, J., Gabbouj, M., and Iosifidis, A. (2018). Subspace support vector data description. In *ICPR*. IEEE. (cited on pages: 112, 114)

[177] Soo Kim, T. and Reiter, A. (2017). Interpretable 3d human action analysis with temporal convolutional networks. In *CVPR Workshops*. (cited on page: 84)

[178] Soomro, K., Zamir, A. R., and Shah, M. (2012). Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*. (cited on pages: 14, 23, 26, 42, and 53)

[179] Srivastava, N., Mansimov, E., and Salakhutdinov, R. (2015). Unsupervised learning of video representations using LSTMs. In *ICML*. (cited on page: 19)

[180] Sultani, W., Chen, C., and Shah, M. (2018). Real-world anomaly detection in surveillance videos. In *CVPR*, pages 6479–6488. (cited on pages: 24, 90, 92, 106, 111, and 112)

[181] Sun, C. and Nevatia, R. (2014). Discover: Discovering important segments for classification of video events and recounting. In *CVPR*. (cited on pages: 46, 72)

[182] Sun, L., Jia, K., Chen, K., Yeung, D.-Y., Shi, B. E., and Savarese, S. (2017). Lattice long short-term memory for human action recognition. In *ICCV*. (cited on page: 84)

[183] Szegedy, C., Ioffe, S., Vanhoucke, V., and Alemi, A. A. (2017). Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*. (cited on pages: 54, 66)

[184] Tax, D. M. and Duin, R. P. (2004). Support vector data description. *Machine learning*, 54(1):45–66. (cited on pages: 87, 91, and 112)

[185] Tian, Y., Krishnan, D., and Isola, P. (2019). Contrastive multiview coding. *arXiv preprint arXiv:1906.05849*. (cited on pages: 69, 71, and 73)

[186] Tian, Y., Pang, G., Chen, Y., Singh, R., Verjans, J. W., and Carneiro, G. (2021). Weakly-supervised video anomaly detection with robust temporal feature magnitude learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4975–4986. (cited on page: 115)

[187] Townsend, J., Koep, N., and Weichwald, S. (2016). Pymanopt: A python toolbox for optimization on manifolds using automatic differentiation. *JMLR*, 17(137):1–5. (cited on page: 106)

[188] Tran, D., Bourdev, L., Fergus, R., Torresani, L., and Paluri, M. (2015). Learning spatiotemporal features with 3d convolutional networks. In *ICCV*. (cited on pages: 4, 11, 19, and 38)

[189] Tran, D., Wang, H., Torresani, L., Ray, J., LeCun, Y., and Paluri, M. (2018). A closer look at spatiotemporal convolutions for action recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 6450–6459. (cited on pages: 5, 19)

[190] Trivedi, M. M., Cheng, S. Y., Childers, E. M., and Krotosky, S. J. (2004). Occupant posture analysis with stereo and thermal infrared video: Algorithms and experimental evaluation. *IEEE Transactions on Vehicular Technology*, 53(6):1698–1712. (cited on page: 90)

[191] Trivedi, M. M., Gandhi, T., and McCall, J. (2007). Looking-in and looking-out of a vehicle: Computer-vision-based enhanced vehicle safety. *IEEE TITS*, 8(1):108–120. (cited on page: 90)

[192] Tsybakov, A. B. et al. (1997). On nonparametric estimation of density level sets. *The Annals of Statistics*, 25(3):948–969. (cited on page: 91)

[193] Tung, F., Zelek, J. S., and Clausi, D. A. (2011). Goal-based trajectory analysis for unusual behaviour detection in intelligent surveillance. *Image and Vision Computing*, 29(4):230–240. (cited on page: 91)

[194] Vahdat, A., Cannons, K., Mori, G., Oh, S., and Kim, I. (2013). Compositional models for video event detection: A multiple kernel learning latent variable approach. In *ICCV*. (cited on page: 46)

[195] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. *arXiv preprint arXiv:1706.03762*. (cited on pages: 20, 68)

[196] Vedaldi, A. and Zisserman, A. (2012). Efficient additive kernels via explicit feature maps. *PAMI*, 34(3):480–492. (cited on page: 52)

[197] Vemulapalli, R., Arrate, F., and Chellappa, R. (2014). Human action recognition by representing 3d skeletons as points in a lie group. In *CVPR*. (cited on pages: 54, 65, and 67)

[198] Wang, H., Kläser, A., Schmid, C., and Liu, C.-L. (2011). Action recognition by dense trajectories. In *CVPR*. (cited on page: 14)

[199] Wang, H., Kläser, A., Schmid, C., and Liu, C.-L. (2013a). Dense trajectories and motion boundary descriptors for action recognition. *IJCV*, 103(1):60–79. (cited on pages: 32, 36, 66, and 83)

[200] Wang, H. and Schmid, C. (2013). Action recognition with improved trajectories. In *ICCV*. (cited on pages: 3, 4, 14, 38, and 58)

[201] Wang, J. and Cherian, A. (2018). Learning discriminative video representations using adversarial perturbations. In *ECCV*. (cited on pages: 7, 89)

[202] Wang, J. and Cherian, A. (2019a). Discriminative video representation learning using support vector classifiers. *IEEE transactions on pattern analysis and machine intelligence*. (cited on pages: 4, 6)

[203] Wang, J. and Cherian, A. (2019b). GODS: generalized one-class discriminative subspaces for anomaly detection. In *ICCV*. (cited on pages: 7, 105)

[204] Wang, J., Cherian, A., and Porikli, F. (2017a). Dynamic pooling for complex event recognition. In *WACV*. (cited on pages: 4, 52)

[205] Wang, J., Cherian, A., and Porikli, F. (2017b). Ordered pooling of optical flow sequences for action recognition. In *WACV*. IEEE. (cited on pages: 6, 14)

[206] Wang, J., Cherian, A., Porikli, F., and Gould, S. (2018a). Video representation learning using discriminative pooling. In *CVPR*. (cited on pages: 6, 73, 81, and 84)

[207] Wang, J. and Torresani, L. (2022). Deformable video transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14053–14062. (cited on pages: 20, 68)

[208] Wang, L., Li, W., Li, W., and Van Gool, L. (2017c). Appearance-and-relation networks for video classification. (cited on page: 19)

[209] Wang, L., Qiao, Y., and Tang, X. (2015). Action recognition with trajectory-pooled deep-convolutional descriptors. In *CVPR*. (cited on pages: 4, 34, 36, and 38)

[210] Wang, L., Xiong, Y., Lin, D., and Van Gool, L. (2017d). Untrimmednets for weakly supervised action recognition and detection. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 4325–4334. (cited on page: 46)

[211] Wang, L., Xiong, Y., Wang, Z., Qiao, Y., Lin, D., Tang, X., and Van Gool, L. (2016). Temporal segment networks: Towards good practices for deep action recognition. In *ECCV*. (cited on pages: 4, 16, 67, 83, and 84)

[212] Wang, T., Chen, J., Zhou, Y., and Snoussi, H. (2013b). Online least squares one-class support vector machines-based abnormal visual event detection. *Sensors*, 13(12):17130–17155. (cited on page: 87)

[213] Wang, X., Girshick, R., Gupta, A., and He, K. (2018b). Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7794–7803. (cited on pages: 19, 20)

[214] Wen, Z. and Yin, W. (2013). A feasible method for optimization with orthogonality constraints. *Mathematical Programming*, 142(1-2):397–434. (cited on page: 99)

[215] Werbos, P. J. (1990). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560. (cited on page: 17)

[216] Wu, J., Yu, Y., Huang, C., and Yu, K. (2015). Deep multiple instance learning for image classification and auto-annotation. In *CVPR*. (cited on page: 45)

[217] Wu, S., Moore, B. E., and Shah, M. (2010). Chaotic invariants of lagrangian particle trajectories for anomaly detection in crowded scenes. In *CVPR*, pages 2054–2060. IEEE. (cited on page: 91)

[218] Wu, Z., Xiong, Y., Yu, S. X., and Lin, D. (2018). Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3733–3742. (cited on pages: 69, 71)

[219] Xie, C., Wang, J., Zhang, Z., Zhou, Y., Xie, L., and Yuille, A. (2017). Adversarial examples for semantic segmentation and object detection. In *ICCV*. (cited on pages: 70, 72)

[220] Xu, D., Ricci, E., Yan, Y., Song, J., and Sebe, N. (2015). Learning deep representations of appearance and motion for anomalous event detection. In *BMVC*. (cited on page: 92)

[221] Xu, D., Yan, Y., Ricci, E., and Sebe, N. (2017). Detecting anomalous events in videos by learning deep representations of appearance and motion. *CVIU*, 156:117–127. (cited on page: 92)

[222] Xu, H., Caramanis, C., and Sanghavi, S. (2010). Robust PCA via outlier pursuit. In *NIPS*, pages 2496–2504. (cited on page: 91)

[223] Yan, X., Zhang, H., Xu, X., Hu, X., and Heng, P.-A. (2021). Learning semantic context from normal samples for unsupervised anomaly detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 3110–3118. (cited on page: 115)

[224] Yi, Y. and Lin, M. (2016). Human action recognition with graph-based multiple-instance learning. *Pattern Recognition*, 53:148–162. (cited on page: 45)

[225] Yu, F. X., Liu, D., Kumar, S., Jebara, T., and Chang, S.-F. (2013). *propto* svm for learning with label proportions. *arXiv preprint arXiv:1306.0886*. (cited on pages: 46, 49, and 50)

[226] Yue-Hei Ng, J., Hausknecht, M., Vijayanarasimhan, S., Vinyals, O., Monga, R., and Toderici, G. (2015). Beyond short snippets: Deep networks for video classification. In *CVPR*. (cited on pages: 4, 19)

[227] Zach, C., Pock, T., and Bischof, H. (2007). A duality based approach for realtime tv-l 1 optical flow. In *Joint Pattern Recognition Symposium*, pages 214–223. Springer. (cited on pages: 31, 59)

[228] Zhang, D., Gatica-Perez, D., Bengio, S., and McCowan, I. (2005). Semi-supervised adapted hmms for unusual event detection. In *CVPR*, pages 611–618. IEEE. (cited on page: 91)

[229] Zhang, D., Meng, D., Li, C., Jiang, L., Zhao, Q., and Han, J. (2015). A self-paced multiple-instance learning framework for co-saliency detection. In *ICCV*. (cited on page: 45)

[230] Zhang, J., Zhang, T., Dai, Y., Harandi, M., and Hartley, R. (2018). Deep unsupervised saliency detection: A multiple noisy labeling perspective. In *CVPR*. (cited on page: 72)

[231] Zhang, P., Lan, C., Xing, J., Zeng, W., Xue, J., and Zheng, N. (2017). View adaptive recurrent neural networks for high performance human action recognition from skeleton data. In *ICCV*. (cited on page: 84)

[232] Zhao, B., Fei-Fei, L., and Xing, E. P. (2011). Online detection of unusual events in videos via dynamic sparse coding. In *CVPR*, pages 3313–3320. IEEE. (cited on pages: 91, 92)

[233] Zhou, Y., Sun, X., Zha, Z.-J., and Zeng, W. (2018). Mict: Mixed 3d/2d convolutional tube for human action recognition. In *CVPR*. (cited on page: 19)

[234] Zhuang, C., Zhai, A. L., and Yamins, D. (2019). Local aggregation for unsupervised learning of visual embeddings. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6002–6012. (cited on pages: 69, 71)