AN ABSTRACT OF THE DISSERTATION OF

Behrooz Mahasseni for the degree of Doctor of Philosophy in Computer Science presented on November 30, 2016.

Title: Robust and Efficient Classification of Videos in the Wild

Abstract approved: ______________________________________

Sinisa Todorovic

Recognizing human actions in videos is a long-standing problem in computer vision with a wide range of applications including video surveillance, content retrieval, and sports analysis. This thesis focuses on addressing efficiency and robustness of video classification in unconstrained real-world settings. The thesis work can be broadly divided into four major parts.

First, we address view-invariant action recognition. This problem is formulated within the multi-task learning framework, where the action model of each viewpoint is specified as a separate task and all tasks are trained jointly.

Second, we address a large-scale action recognition in uncontrolled settings. For robustness, we augment the standard training video dataset with additional data from another modality data source – namely, 3D skeleton sequences of human body motion –. A recurrent neural network called long short-term memory (LSTM) is used to encode sequences from 3D skeleton data. For learning another LSTM for video classification, we use a modified hybrid backpropagation through time algorithm.

Third, we address the unsupervised video summarization. We formulate the problem as a subset frame selection and specified a novel deep generative network to compute a video summary with the smallest representation error.

Fourth, we introduce the new problem of budget-aware semantic segmentation of videos. In this line of work, we consider two models. The first model uses a conditional random field (CRF) model and replaces the standard inference steps for feature computation with a sequential policy which intelligently selects a subset of regions and their corresponding features. The second model is a deep recurrent policy which is learned to select a subset of frames and uses a shallow convolutional neural network (CNN) to propagate the available segmentation to unlabeled frames.

This research has advanced the state of the art in computer vision because the approaches developed enabled meeting stringent runtime requirements arising in many applications, and working in less sanitized settings.

Robust and Efficient Classification of Videos in the Wild

by

Behrooz Mahasseni

A DISSERTATION

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Doctor of Philosophy

Presented November 30, 2016
Commencement June 2017

Doctor of Philosophy dissertation of Behrooz Mahasseni presented on November 30, 2016.

APPROVED:

Major Professor, representing Computer Science

Director of the School of Electrical Engineering and Computer Science

Dean of the Graduate School

I understand that my dissertation will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my dissertation to any reader upon request.

Behrooz Mahasseni, Author

ACKNOWLEDGEMENTS

First of all, I would like to thank my advisor, Professor Sinisa Todorovic, for his support during my Ph.D. study. His advice and guidance have been invaluable both for pursuing my Ph.D. and a successful career. He indeed helped me beyond just developing technical skills but improving my intellectual thinking and presentation skills which are necessary for a long-term success.

Next, I would like to thank Prof. Alan Fern for his excellent help during my Ph.D. I learned a lot from him on multiple projects we collaborated on. I also thank all my previous teachers and professors and my Ph.D. committee.

I also would like to thank my friends and colleagues in the computer vision research group at OSU for our technical discussions, continuous brainstorming and funny moments. In addition, I would like to thank Jan Kautz, Xiaodong Yan, Alok Govil, and Gousev Evgani for the six insightful and challenging months at the NVIDIA Research and Qualcomm Corp R&D.

I am also grateful to my good friends and colleagues, in university and my internships, Anirban Roy, Michael Lam, Sheng Chen, Peng Lei, Christos Margiolas, Alberto Garcia, Suren Jayasuriya and Abhishek Badki.

Finally, I would like to thank my family for their unconditional love and endless support. I dedicate this thesis to my mom for nursing me with affection and for her endless love, to my dad whose words of encouragement ring in my ears, to my wife for her endless support and advice, to my brother who has never left my side and is very special, and finally to my son which his presence has filled my heart with joy.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF TABLES

LIST OF TABLES (Continued)

LIST OF ALGORITHMS

I dedicate this thesis to my mom for nursing me with affection and for her endless love, to my dad whose words of encouragement ring in my ears, to my wife for her endless support and advice, to my brother who has never left my side and is very special, and finally to my son which his presence has filled my heart with joy.

Chapter 1: Introduction

Since the introduction of low-cost video recording devices such as mobile phones, wearable and ego-centric cameras and surveillance equipment, the volume of the video data has been rapidly increasing. The forecast report [59] shows more than 80% of the Internet content will be video data in 2020. Statistical analysis of the upload rate of the video data [3] also supports this observation. To analyze, browse and search today‘s ever-growing video collections, we need robust algorithms that understand and summarize the video content in an efficient way. In this thesis we focus on improving robustness and efficiency of the state of the art on ‘large-scale action recognition’, ‘unsupervised video summarization’, and ‘semantic video segmentation’, where robustness is defined in terms of the model‘s accuracy and efficiency is defined in terms of the inference processing time.

Understanding videos of human actions, recorded in an uncontrolled setting, is an open problem in computer vision. First, these videos are captured in-the-wild, where the actions are viewed from various camera viewpoints and distances, and under partial occlusion. Second is a large number of action classes and the granularity of the action categories (e.g. freestyle BMX, cyclocross and road bicycle racing are similar actions which all contain humans riding a bike(s)). Third, the actions may be performed by individuals or groups of people (e.g., skateboarding vs. marathon), and may be characterized by critical human-human and human-object interactions (e.g., bull-riding vs. horseback riding). Finally, the same action can be performed with different motion styles and variations in speed.

Action recognition has been a long-standing problem in computer vision [180, 177, 159, 133, 104, 141, 126, 115, 161, 66, 46, 10, 63, 144, 72, 184, 25, 158, 134]. We advanced the state of the art by learning how to be robust and efficient as opposed to common heuristics. We particularly proposed a multimodal deep learning approach where multiple models which are grounded on different input sources are trained jointly.

Video summarization has attracted recent attention with the appearance of large data. Given a long video, the goal is to provide a shorter length video which still provides informative content regarding the original video. One difficulty is that there is no unique way to define the video summarization problem and the exact definition highly depends on the application [127, 120, 121, 7, 164, 139, 44, 89, 79, 65, 118, 56]. More importantly, we believe that the main challenge is the inherent subjectiveness of a ‘good’ summary.

Finally, the design of any practical computer vision system is typically informed by a trade-off between efficiency and accuracy. Good computational efficiency is usually achieved by taking a number of heuristic pre- and post-processing steps and integrating them with the main approach. For example, vision practitioners heuristically limit the types of features to be extracted (e.g., low-cost ones) from an image or a video. While these steps have been satisfactory for small-scale problems, their heuristic nature makes an adaptation of existing systems to settings with stringent run-time requirements very difficult.

In the following, we provide an outline of this thesis. This thesis is a manuscript style.

Chapters 2 and 3 present our work on human action recognition. We hypothesize that robustness can be achieved through utilizing multiple different input modalities. The modalities are not necessarily from different physical sensors (e.g. videos captured from different viewpoints can be considered as distinct input modalities).

Our work on robust action recognition under varying viewpoints which is formulated as a novel latent multi-task learning is described in chapter 2. This work follows the state of the art traditional paradigm in computer vision. The common practice in the traditional paradigm is to represent a video by a summary of locally computed appearance or motion feature descriptors [20, 21] for a set of informative interest points [24, 86, 167, 156], or a dense set of feature points sampled in space-time [154, 155]. A probabilistic model is then used [85, 5, 84, 140, 98] to classify the action.

A historic paradigm shift happened in machine learning research by the recent breakthrough in deep learning and the concept of big data. In this new practice, the emphasis is on end-to-end learning of feature hierarchies. Deep models have shown impressive accuracy in many vision problems including action classification from wild videos [10, 63, 144, 72, 184, 25, 158, 134, 125, 138]. The current research efforts to improve deep action classification models can be grouped as follows: (a) Increasing the amount of training data [72] or using additional complementary input sources [111, 62] (b) Fusing hand-designed and deep-convolutional features [158, 134, 184], and (c) Combining Convolutional Neural Networks (CNNs) with either graphical models [144, 55, 112], or recurrent neural networks [25, 112] for capturing complex temporal dynamics. Despite these efforts, the action classification accuracy of existing work is still markedly below the state-of-the-art performance on the related large-scale problem of image classification. On the other hand, although the feed-forward architecture of these models provides a faster inference in images, efficiency is still an important practical challenge in videos. This motivates us to seek a novel strategy to improve efficiency and robustness of deep action classification models. We propose a novel regularization, which uses complementary information from 3D skeleton data during training to improve the classification accuracy. This work is presented in chapter 3

In chapter 4 we formulate the video summarization as an unsupervised subset selection problem, i.e. given a complete set of video frames, the key goal is to select a subset of those frames that preserve most of the important content of the original video frames. Our main contribution is to introduce a completely different view on the definition of a 'good summary'. Instead of the standard approach, which requires a heuristic, human engineered metric or measures to summarize a video, we learn both the metric and the summarization algorithm.

In chapters 5 and 6 we study on the efficiency of semantic video segmentation. Good computational efficiency is usually achieved by taking a number of heuristic pre- and post-processing steps and integrating them with the main approach. For example, vision practitioners heuristically limit the types of features to be extracted (e.g., low-cost ones), as well as locations and scales in images and video from which they are extracted. While these steps have been satisfactory for small-scale problems, their heuristic nature makes an adaptation of existing systems to settings with stringent runtime requirements very difficult.

The trade-off between efficiency and accuracy can be seen as a cost-aware inference. The cost-aware inference is a more challenging problem compared to the standard recognition tasks. The cost-aware inference can be formulated as a decision-making problem. As an example, consider the semantic video segmentation which is a well-known problem in computer vision community. While the accuracy of a semantic video segmentation algorithm is defined in terms of the correct estimated pixel labels, the efficiency is defined in terms of the processing time (budget) the algorithm needs to provide pixel labeling.

However, none of the current solutions [102, 113, 73, 182, 18] satisfy the runtime expectations. Only a few heuristic approaches have considered efficiency in semantic segmentation [61, 95, 50]. Instead, we define the new problem of 'budget-aware semantic video segmentation'. We hypothesize that casting the budget-aware semantic video segmentation inference as a Markov Decision Process (MDP), allows us to provide a principled framework for considering the trade-off between accuracy and efficiency. While in chapter 5 we learn the policy through standard classification-based policy iteration, in chapter 6 we train the policy, modeled as a recurrent neural network, using recurrent policy gradient approach.

Due to the diversity of prior work on each topic we choose to provide a complete review of the prior work on each topic at the beginning of the respective chapter.

Chapter 2: Latent Multitask Learning for View-Invariant Action Recognition

Abstract

This paper presents an approach to view-invariant action recognition, where human poses and motions exhibit large variations across different camera viewpoints. When each viewpoint of a given set of action classes is specified as a learning task then multitask learning appears suitable for achieving view invariance in recognition. We extend the standard multitask learning to allow identifying: (1) latent groupings of action views (i.e., tasks), and (2) discriminative action parts, along with joint learning of all tasks. This is because it seems reasonable to expect that certain distinct views are more correlated than some others, and thus identifying correlated views could improve recognition. Also, part-based modeling is expected to improve robustness against self-occlusion when actors are imaged from different views. Results on the benchmark datasets show that we outperform standard multitask learning by 21.9%, and the state-of-the-art alternatives by 4.5–6%.

2.1 Introduction

This paper considers the problem of view-invariant action recognition. Given a video that shows a human action (e.g., walking, jumping), we want to identify the action class and camera viewpoint. The videos are captured from different camera viewpoints, which are taken to be discrete and indexed by the viewpoint identifier, or viewpoint, for short.

Invariance to viewpoint changes is critical for action recognition, because people's motion trajectories may take arbitrary directions relative to the camera viewpoint while performing an action. In our setting, natural variations of action instances within a class are augmented by variations in their appearance across different viewpoints.

One way to achieve view invariance could be to reason about a 3D layout of the scene, or 3D volume of the human body, so that the video features can be adapted from one view to another through geometric transformations [180, 177, 159, 133, 104]. However, this framework critically depends on accurate detection of the body joints and contour, which are still open problems in real-world settings. An alternative way would be to extract view-invariant video features [141, 126, 115, 161, 66, 46]. Some of these methods are limited by requiring access to motion capture data, while others find invariant features for only a subset of views.

The third group of approaches use knowledge transfer. They seek to extend knowledge acquired in training from one or a limited number of views to other target views where recognition will be performed. They either transform view-dependent video features to a new view-invariant feature space [91, 99], or adapt model parameters to the target views [30, 31, 173]. The transformation is learned on the co-occurrence statistics of view-dependent features. This is attractive, because knowledge transfer relaxes the requirements for accurate 3D scene and 3D human-body reconstruction. However, these approaches require access to simultaneous multiview observations of the same action instance (except for [91]). In addition, they represent a video by a bag-of-words (BoW) disregarding the layout of human-body parts. Accounting for body parts seems important in our setting, because they are subject to self-occlusion when imaged from different views.

To approach our problem, we specify that each viewpoint of a given set of action classes is a learning task. Then, view invariance in recognition could be achieved by jointly learning all the tasks using Multitask Learning (MTL) [14]. This is because MTL would be able to estimate a latent feature representation shared across all views. While MTL is well known to vision [146, 101, 123, 187], it has never been used for view-invariant action recognition.

MTL is based on the assumption that all the tasks considered (i.e., in our case viewpoints of actions) are correlated. However, in our setting, this assumption may be too strong. Human actions may occur in cluttered scenes, and discriminative movements of the human body may not be visible from all viewpoints. Therefore, MTL is bound to under- or over-estimate the correlation among all the viewpoints, due to confusing background and foreground features, and disregarding self-occlusions of the human body.

To address the above issues, we extend the standard MTL to Latent Multitask Learning (LMTL). Our LMTL uses a part-based action representation, instead of the standard BoW. In this way, LMTL is enabled to identify foreground video features which group into discriminative action parts, each corresponding to characteristic movements of a human-body part. In addition, LMTL is enabled to identify latent groupings of correlated viewpoints of a given set of action classes. Thus, LMTL learns a new shared feature space, such that each group of camera viewpoints found to be correlated are allowed to share features, whereas this sharing is prohibited between the groups.

We use the latent large-margin framework [181] to formulate LMTL, wherein we incorporate the mixed integer programming of [68] for grouping the viewpoints. This extends the work of [68], which does not account for latent parts. Within each group of viewpoints, a shared feature representation is estimated and used for learning parameters of a part-based action model, subject to the trace-norm regularization. Fig. 2.1 shows an overview of our approach.

In the sequel, Sec. 2.2 gives a more formal overview of our approach, Sec. 2.3 reviews the standard MTL, Sec. 2.4 formulates our LMTL, Sec. 2.5 specifies inference, Sec. 2.6 describes video features, and Sec. 2.7 presents our results.

2.2 Overview of Our Approach

Our LMTL framework leverages a deformable parts model (DPM) [33]. DPM has K nodes representing K action parts, connected in a star structure. An action part is a discriminative space-time window in the video volume. Thus, each node of DPM can be characterized by spatiotemporal features extracted from the corresponding space-time window. The nodes are connected to the root, where the graph edges encode space-time deformations of the action parts. DPM parameters represent weights associated with nodes and edges. The weights can be learned within the latent large-margin framework [181], aimed at jointly discovering the K latent parts, and estimating their weights so as to maximize the discriminativeness of DPM across a set of action classes. Below, we give an overview of how to ground action parts onto raw pixels, and how to perform view-invariant action recognition.

Access to action parts is provided by representing a video by a large set of overlapping space-time windows of different sizes and shapes, $\mathbb{V} = \{\mathcal{V}_1, ..., \mathcal{V}_N\}$. When an action occurs in the video, it occupies only a subset of $K \ll N$ windows, $\mathbb{A} = \{\mathcal{V}_k : \mathcal{V}_k \in \mathbb{V}, k = 1, ..., K\}$, corresponding to the K action parts. Video features extracted from $\mathbb{A}$, and spatiotemporal displacements of the windows in $\mathbb{A}$ can be represented by a d-dimensional vector, $\boldsymbol{\phi}(\boldsymbol{x}, y, \boldsymbol{h})$, where $\boldsymbol{x}$ denotes the features; y is the action class; and $\boldsymbol{h}$ denotes the space-time locations of the windows in $\mathbb{A}$. For a set of M action classes, we learn a multiclass DPM by using an augmented D-dimensional feature vector, $\boldsymbol{\Phi}(\boldsymbol{x}, y, \boldsymbol{h})$, where $D = d \cdot M$. $\boldsymbol{\Phi}(\boldsymbol{x}, y, \boldsymbol{h})$ is a sparse vector, whose all elements are set to zero, except in the yth segment of d elements copied from $\boldsymbol{\phi}(\boldsymbol{x}, y, \boldsymbol{h})$.

Our LMTL is aimed at transforming the input view-dependent $\boldsymbol{\Phi}(\boldsymbol{x}, y, \boldsymbol{h})$ to a new, view-invariant feature space. We use a linear transform, $U^{\top}\boldsymbol{\Phi}(\boldsymbol{x}, y, \boldsymbol{h})$, where $U \in \mathbb{R}^{D\times D}$ is an

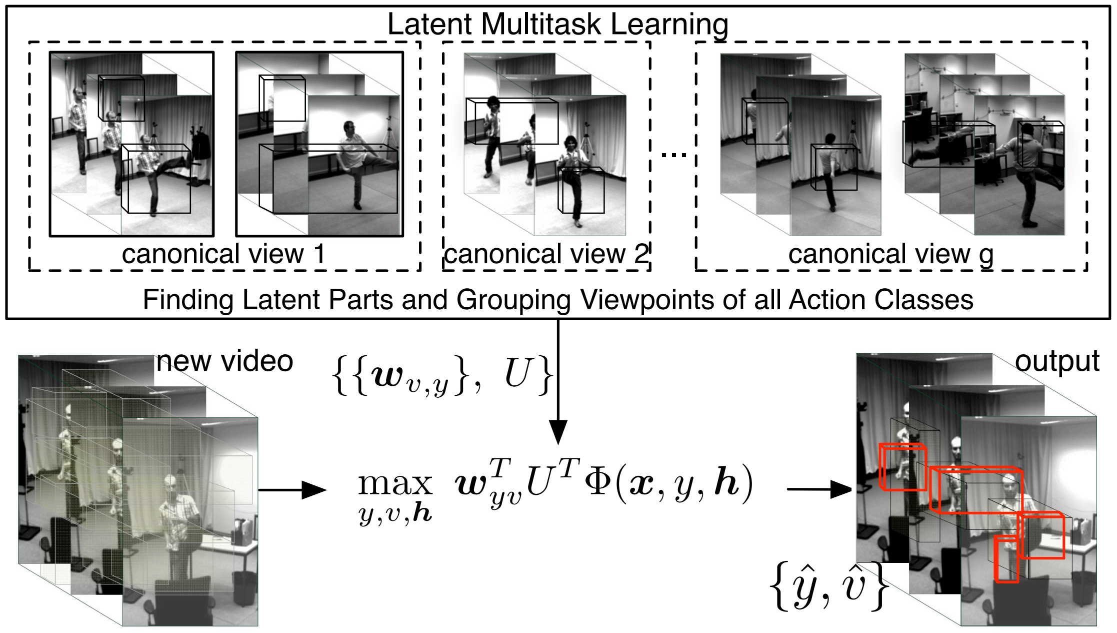

Latent Multitask Learning
canonical view 1
canonical view 2
canonical view g
Finding Latent Parts and Grouping Viewpoints of all Action Classes
new video
$\{\{\boldsymbol{w}_{v,y}\},\ U\}$
$\max_{y,v,\boldsymbol{h}} \boldsymbol{w}_{yv}^T U^T \Phi(\boldsymbol{x}, y, \boldsymbol{h})$
output
$\{\hat{y}, \hat{v}\}$

Figure 2.1: For a given video, we estimate the action class $\hat{y}$ and viewpoint $\hat{v}$ using Latent Multitask Learning (LMTL). LMTL identifies action parts, $\boldsymbol{h}$, and groups of correlated camera viewpoints. LMTL learns a linear transformation U to map input view-dependent features to a new feature space, partitioned into subspaces which are shared by viewpoints within the same group.

orthogonal square matrix. For recognition, we define the multiclass discriminant function F as

$$F_{y,v,\boldsymbol{h}}(\boldsymbol{x}) = \boldsymbol{w}_v^\top U^\top \boldsymbol{\Phi}(\boldsymbol{x}, y, \boldsymbol{h}), \tag{2.1}$$

where v is the viewpoint, and $\boldsymbol{w}_v \in \mathbb{R}^D$ are the multiclass DPM parameters. Given a video, the action class and viewpoint are estimated via localizing latent action parts as

$$(\hat{y}, \hat{v}) = \arg\max_{y,v,\boldsymbol{h}} \; F_{y,v,\boldsymbol{h}}(\boldsymbol{x}). \tag{2.2}$$

We learn $\boldsymbol{w}_v$ using LMTL. In the following section, we briefly review the standard MTL, and defer the specification of LMTL to Sec. 2.4.

2.3 A Brief Review of Multitask Learning

The model parameters $\boldsymbol{w}_v$, defined in Sec. 2.2, can be learned using MTL, where each task v represents recognition of one of M action classes imaged from the given view v. This section, first, specifies a learning paradigm where the tasks are learned independently, referred to as Baseline 1. Then, we present the standard MTL, referred to as Baseline 2. Finally, we review the recent task-grouping MTL of [68], referred to as Baseline 3. These baselines are used in our experiments for comparison (Sec. 2.7).

2.3.1 Baseline 1 = Learning Independent Tasks

Let W be a matrix whose columns are $\boldsymbol{w}_v$, indexed by viewpoints $v = 1, ..., V$. Also, let $\Delta(y, \hat{y}(\boldsymbol{w}_v, \boldsymbol{x}))$ denote a loss function of recognizing action class $\hat{y}$ when the true class is y in the vth task. For this baseline, we assume that all tasks use the same feature space, with feature vectors $\boldsymbol{x}$. Given training data $\mathcal{D}_v = \{(\boldsymbol{x}_i, y_i) : i = 1, 2, ...\}$, the tasks can be learned independently as

$$\min_W \textstyle\sum_{v=1}^{V} \sum_{(\boldsymbol{x}_i, y_i) \in \mathcal{D}_v} \Delta(y_i, \hat{y}(\boldsymbol{w}_v, \boldsymbol{x}_i)) + \gamma \|W\|_F^2, \tag{2.3}$$

where $\|W\|_F^2 = \sum_v \|\boldsymbol{w}_v\|_2^2$ is the Frobenius norm of W.

2.3.2 Baseline 2 = MTL

MTL extends Baseline 1 by finding a common feature subspace on which all the tasks perform well. We use a linear transformation U of the original features, $U^\top \boldsymbol{x}$, to find a lower-dimensional subspace. As in [68], we regularize MTL learning of every $\boldsymbol{w}_v$ by using the (2,1)-norm of W, $\|W\|_{2,1} = \sum_{i=1}^{D} \sqrt{\sum_v w_{i,v}^2}$, as

$$\min_{W,U} \textstyle\sum_{v=1}^{V} \sum_{(\boldsymbol{x}_i, y_i) \in \mathcal{D}_v} \Delta(y_i, \hat{y}(\boldsymbol{w}_v, U^\top \boldsymbol{x}_i)) + \gamma \|W\|_{2,1}^2. \tag{2.4}$$

The regularization of W in (2.4) enforces row-sparseness of W, i.e., maximizes the number of zero rows in W.

Note that the loss in (2.4) is not convex with respect to both W and U; but it is convex with respect to each of them separately. For solving (2.4), we use the approach of [8], where a similar optimization problem is addressed by removing the non-convex dependency of the loss function on two variables. To this end, we introduce the following notation:

$$(\Theta,\ \Omega) = (UW,\ \ U\text{diag}(\boldsymbol{\lambda})U^\top), \tag{2.5}$$

where $\boldsymbol{\lambda} = [\frac{\|W_1\|_2}{\|W\|_{2,1}}, ..., \frac{\|W_i\|_2}{\|W\|_{2,1}}, ..., \frac{\|W_D\|_2}{\|W\|_{2,1}}]$ and W_i is the ith row of W. The columns of matrix Θ are denoted as $\{\boldsymbol{\theta}_v : v = 1, ..., V\}$. Using this new notation, the minimization problem of (2.4) can be expressed as (see the proof in [8]):

$$\min_{\Theta,\Omega} \sum_v \sum_{(\boldsymbol{x}_i, y_i) \in \mathcal{D}_v} \Delta(y_i, \hat{y}(\boldsymbol{\theta}_v, \boldsymbol{x}_i)) + \gamma \sum_v \boldsymbol{\theta}_v^\top \Omega^{-1} \boldsymbol{\theta}_v. \tag{2.6}$$

From (2.1) and (2.2), our inference requires the computation of the product $\Theta = UW$, rather than using the matrices U and W individually. Therefore, instead of learning U and W, it suffices to learn Θ, and then directly use Θ in (2.2). Next, we describe an algorithm for estimating Θ and Ω from (2.6).

The following two-step iterative algorithm, presented in [8], can be used for solving the optimization problem of (2.6):

1. Given Θ, find Ω from (2.6). By theorem 4.1 in [8], there is a closed-form solution $\Omega = \frac{(\Theta\Theta^\top)^{\frac{1}{2}}}{\text{Trace}((\Theta\Theta^\top)^{\frac{1}{2}})}$.
2. Given Ω, find Θ from (2.6). Substituting the closed-form solution of Ω from step 1 in (2.6),

we get $\min_{\Theta} \sum_v \sum_{(\boldsymbol{x}_i,y_i)\in\mathcal{D}_v} \Delta(y_i, \hat{y}(\boldsymbol{\theta}_v, x_i)) + \gamma\|\Theta\|_*^2$, where $\|\cdot\|_*$ is the trace norm.

2.3.3 Baseline 3 = Task-grouping MTL

In [68], the standard MTL is extended by grouping tasks and finding feature subspaces for different groups. As mentioned in Sec. 2.1, the approach of [68] is agnostic of parts, and thus is our Baseline 3. The task grouping can be achieved by separating the regularization of Θ over distinct task groups in (2.6) as

$$\min_{\Theta,\Omega} \sum_v \sum_i \Delta(y_i, \hat{y}(\boldsymbol{\theta}_v, \boldsymbol{x}_i)) + \gamma \sum_g \sum_{v_g} \boldsymbol{\theta}_{v_g}^\top \Omega^{-1} \boldsymbol{\theta}_{v_g}, \tag{2.7}$$

where v_g denotes viewpoints that belong to group g. Note that a solution of (2.7) needs to resolve the latent assignment of viewpoints to groups, in addition to finding Θ and Ω.

As for Baseline 2, we can use the above two-step iterative algorithm for solving (2.7). For given Θ, the first step finds the closed-form solution $\Omega = \frac{(\Theta\Theta^\top)^{\frac{1}{2}}}{\text{Trace}((\Theta\Theta^\top)^{\frac{1}{2}})}$. Then, in the second step, we substitute this closed-form solution for Ω into (2.7), and obtain the following simpler problem:

$$\min_{\Theta} \sum_v \sum_i \Delta(y_i, \hat{y}(\boldsymbol{\theta}_v, x_i)) + \gamma \sum_g \|\Theta_g\|_*^2. \tag{2.8}$$

where Θ_g is a matrix whose columns are parameters $\boldsymbol{\theta}_{v_g}$ of the viewpoints in group g.

Mixture integer programming can be used in (2.8) to identify the latent assignment of viewpoints to groups. Let $Q_g \in \mathbb{R}^{V\times V}$ denote the diagonal assignment matrices for grouping the viewpoints into their respective groups g. Thus, (2.8) can be conveniently expressed as

$$\begin{aligned} &\min_{\Theta,\{Q_g\}} \ \sum_v \sum_i \Delta(y_i, \hat{y}(\boldsymbol{\theta}_v, x_i)) + \gamma \sum_g \|\Theta Q_g\|_*^2 \\ &\text{s.t.} \ \sum_g Q_g = I, \end{aligned} \tag{2.9}$$

where I is the identity matrix. Note that when the maximum $g = 1$, Baseline 3 is equivalent to Baseline 2.

2.4 Latent Multitask Learning

This section specifies our LMTL. We extend the task-grouping MTL of [68] (i.e., Baseline 3) to additionally identify discriminative action parts. The goal of LMTL is to learn Θ and Ω, defined in (2.5), and the viewpoint grouping matrices $\{Q_g\}$, defined in (2.9), and use them for inference in (2.2). Below, we first specify how to estimate Ω, then formulate a new optimization problem for estimating Θ, and $\{Q_g\}$, and finally present an iterative algorithm for learning all LMTL parameters Θ, Ω, and $\{Q_g\}$ on training data.

As in Baselines 2 and 3, we can readily estimate Ω, given Θ, using the closed-form solution $\Omega = \frac{(\Theta\Theta^\top)^{\frac{1}{2}}}{\text{Trace}((\Theta\Theta^\top)^{\frac{1}{2}})}$.

2.4.1 New Optimization Problem.

For learning Θ, and $\{Q_g\}$, we introduce a new loss function, and substitute it in (2.9). Let $\boldsymbol{h}^*_{vi} = \boldsymbol{h}^*(\boldsymbol{\theta}_v, \boldsymbol{x}_i)$ denote the estimates of vth task for latent action parts in a training video, $(\boldsymbol{x}_i, y_i) \in \mathcal{D}_v$, given its true action class y_i. Also, let $\hat{\boldsymbol{h}}_{vi} = \hat{\boldsymbol{h}}(\boldsymbol{\theta}_v, \boldsymbol{x}_i)$ denote the estimates of vth task for latent action parts in the same training video, $(\boldsymbol{x}_i, y_i) \in \mathcal{D}_v$, without the knowledge of its true action class, but given some estimate $\hat{y}_{vi} = \hat{y}(\boldsymbol{\theta}_v, \boldsymbol{x}_i)$. Then, as in [33, 181], we define a loss function, $\Delta(y_i, \boldsymbol{h}^*_{vi}, \hat{y}_{vi}, \hat{\boldsymbol{h}}_{vi})$, in terms of both action class labels and latent variables, and substitute it in (2.9). Thus, we obtain the following new optimization problem:

$$\begin{aligned} &\min_{\Theta,\{Q_g\}} \; \textstyle\sum_v \sum_i \Delta(y_i, \boldsymbol{h}^*_{vi}, \hat{y}_{vi}, \hat{\boldsymbol{h}}_{vi}) + \gamma \sum_g \|\Theta Q_g\|^2_* \\ &\text{s.t.} \;\; \textstyle\sum_g Q_g = I, \end{aligned} \tag{2.10}$$

The above loss function $\Delta(y_i, \boldsymbol{h}^*_{vi}, \hat{y}_{vi}, \hat{\boldsymbol{h}}_{vi})$ is not convex. As discussed in [181, 33], defining a loss function based on h^* is difficult. Following the derivation in [181, 33], we approximate $\Delta(y_i, \boldsymbol{h}^*_{vi}, \hat{y}_{vi}, \hat{\boldsymbol{h}}_{vi})$ with the upper-bound loss function, $\Delta(y_i, \boldsymbol{h}^*_{vi}, \hat{y}_{vi}, \hat{\boldsymbol{h}}_{vi}) \leq \Delta_{\text{ub}}(y_i, \hat{y}_{vi}, \hat{\boldsymbol{h}}_{vi})$, where

$$\begin{aligned} \Delta_{\text{ub}}(y_i, \hat{y}_{vi}, \hat{\boldsymbol{h}}_{vi}) = &\max_{\hat{y},\hat{\boldsymbol{h}}} [\boldsymbol{\theta}_v{}^\top U^\top \boldsymbol{\Phi}(\boldsymbol{x}_i, \hat{y}, \hat{\boldsymbol{h}}) + \Delta_{01}(y_i, \hat{y})] \\ &- \max_{h} [\boldsymbol{\theta}_v{}^\top U^\top \boldsymbol{\Phi}(\boldsymbol{x}_i, y_i, \boldsymbol{h})], \end{aligned} \tag{2.11}$$

where $\Delta_{01}(y_i, \hat{y})$ is defined as the standard zero-one loss, taking value 1 when $y_i \neq \hat{y}$, and 0, otherwise. We substitute Δ_{ub} in (2.10), which gives our final formulation for learning Θ and Q_g parameters:

$$\begin{aligned} &\min_{\Theta,\{Q_g\}} \ \textstyle\sum_v \sum_i \Delta_{\text{ub}}(y_i, \hat{y}_{vi}, \hat{\boldsymbol{h}}_{vi}) + \gamma \sum_g \|\Theta Q_g\|_*^2 \\ &\text{s.t.} \ \textstyle\sum_g Q_g = I. \end{aligned} \tag{2.12}$$

2.4.2 Iterative Algorithm.

Given training data, $\mathcal{D}_v = \{(\boldsymbol{x}_i, y_i)\}$, $v = 1, ..., V$, the parameters of LTML, Θ, Ω and $\{Q_g\}$, are learned using an iterative algorithm, where each iteration consists of the following three steps.

Step 1: Given Ω and $\{Q_g\}$, find Θ from (2.12). The key ideas is that for given $\{Q_g\}$, the optimization problem of (2.12) is separable, i.e., the columns of Θ, $\{\boldsymbol{\theta_v} : v = 1, ..., V\}$, can be independently estimated. This follows from $\sum_g \|\Theta Q_g\|_*^2 = \sum_g \|\Theta_g\|_*^2 = \sum_g \sum_{v \in g} \|\boldsymbol{\theta}_v\|_2^2$, where $v \in g$ means that the latter sum is only over those viewpoints in the group g. From (2.11) and (2.12), we derive the following V optimization problems for finding Θ:

$$\begin{aligned} \min_{\boldsymbol{\theta}_v} \Big[\frac{1}{2} \|\boldsymbol{\theta_v}\|^2 + C \sum_{i \in \mathcal{D}_v}^{n} \max_{\hat{y}, \hat{\boldsymbol{h}}} [\boldsymbol{\theta_v} \boldsymbol{\Phi}(\boldsymbol{x}_i, \hat{y}, \hat{\boldsymbol{h}}) + \Delta_{01}(y_i, \hat{y})] \\ -C \sum_{i \in \mathcal{D}_v}^{n} \max_{\boldsymbol{h}} [\boldsymbol{\theta_v} \boldsymbol{\Phi}(\boldsymbol{x}_i, y_i, \boldsymbol{h})] \Big], \end{aligned} \tag{2.13}$$

Note that (2.13) is the standard latent structured SVM formulation [33, 181], and can be efficiently solved using the CCCP algorithm, as in [33, 181].

Step 2: Given $\{Q_g\}$ and Θ, compute Ω using the closed-form solution, $\Omega = \frac{(\Theta\Theta^\top)^{\frac{1}{2}}}{\text{Trace}((\Theta\Theta^\top)^{\frac{1}{2}})}$.

Step 3: Given Θ and Ω, find $\{Q_g\}$. In this step, we use the gradient descent, following the derivation presented in [68]. The gradient decent of the Lagrangian function of (2.12) is made possible in [68] by relaxing the integer regularization in (2.12) to $\sum_g \|\Theta\sqrt{Q_g}\|_*^2$. For binary solutions of Q_g the relaxed regularization is equivalent to the original one.

2.5 Inference

Given a set of space-time windows of a new video, $\mathbb{V} = \{\mathcal{V}_1, ..., \mathcal{V}_N\}$, inference consists of two steps.

In the first step, we infer latent variables, $\hat{\boldsymbol{h}}$, i.e., identify K action parts in $\mathbb{V}$, using the distance transform accommodated for 2D+time volumes [33]. From (2.1), this uniquely specifies the augmented feature vector $\boldsymbol{\Phi}(\boldsymbol{x}, y, \hat{\boldsymbol{h}})$ that is used for computing the multiclass discriminant function:

$$F_{y,v,\hat{\boldsymbol{h}}}(\boldsymbol{x}) = \boldsymbol{w}_v^\top U^\top \boldsymbol{\Phi}(\boldsymbol{x}, y, \hat{\boldsymbol{h}}) = \boldsymbol{\theta}_v^\top \boldsymbol{\Phi}(\boldsymbol{x}, y, \hat{\boldsymbol{h}}). \tag{2.14}$$

In the second step, from (2.2) and (2.14), we recognize the action class and viewpoint of the new video as

$$(\hat{y}, \hat{v}) = \arg\max_{y,v} \ \boldsymbol{\theta}_v^\top \boldsymbol{\Phi}(\boldsymbol{x}, y, \hat{\boldsymbol{h}}). \tag{2.15}$$

2.6 Features

This section describes our feature vectors $\boldsymbol{x}$ and $\boldsymbol{\phi}(\boldsymbol{x}, y, \hat{\boldsymbol{h}})$.

A video is represented by a large set of overlapping space-time (2D+t) windows of different sizes. Each 2D+t window is characterized by the standard BoW with 2000 codewords. For extracting the codewords, we use dense trajectories [151], which have shown promise for view-invariant action recognition [173]. The dense trajectories are described by a concatenation of the following descriptors: trajectory(30), HOG(108), HOF(96), MBHx(96), and MBHy(96)). For extracting the dense trajectories, and computing their descriptors, we use the software implementation from [152]. Given the set of feature descriptors from all videos in training data, we use K-means to find the 2000 codewords, and thus produce the BoW descriptions of all space-time windows in the video.

$\boldsymbol{\phi}(\boldsymbol{x}, y, \hat{\boldsymbol{h}})$ is formed by concatenating unary and pairwise potentials of action parts. The unary potential is a BoW associated with the corresponding space-time window. The pairwise potential is defined as the Euclidean distance between the two closest corners of the 2D+t windows corresponding to the root and the action part.

2.7 Results

Datasets. We evaluate our approach on three benchmark datasets including: the IXMAS dataset [161], the newer version of IXMAS dataset [160] referred to as IXMAS(new), and the i3DPost multiview human action dataset [43]. IXMAS has 12 different actions performed by 11 actors three times. These actions have been recorded in five different viewpoints. IXMAS(new) has the same set of actions as IXMAS, recorded from five different viewpoints with different cameras. Two-thirds of the videos contain objects in the scene partially occluding the actors. i3DPost has 13 actions of 8 people recorded from 8 viewpoints. In addition to simple actions (e.g. Walk, Run, Bend), i3DPost contains structured actions (e.g. Run-Fall, Run, Jump, Walk), and actions with two actors (e.g. Pull). Prior work reports accuracy only for the simple actions of i3DPost. We evaluate our approach on all the actions of i3DPost.

Video Representation. A video is split into overlapping 2D+t windows of varying width, height and time duration. The width and height vary in a range of [100, 200] pixels, and time varies in [5-90] frames in increments of 10 frames. This generates approximately 25000 overlapping 2D+t windows for a video of 90 frames of size (400×300) pixels. In our experiments, our approach is relatively insensitive to the size parameters and placement of 2D+t windows. For example, a twice larger size and coarser placement of 2D+t windows, totaling 10000, yields on average a performance reduction by 2%.

Input parameters to our LMTL include: the number of action parts $K = \{2, 4, 6, 8\}$ and the number of groups $g \in \{1, 5, 6, 7, 8\}$. We test our sensitivity to the specific choice of K and g. Fig. 2.2 shows that changes of g affect our average accuracy on the action classes of i3DPost. Varying K in the subrange $4 - -6$ seems to have negligent effect on our average accuracy. In the following, we will use $g = 3$ and $K = 6$, which give the best results, if not mentioned otherwise.

Baselines. Baseline 1 learns action classifiers separately for different viewpoints, and is specified in (2.3). Baseline 2 learns a common feature subspace for all tasks; it is introduced in [8], and specified in (2.4). Baseline 3 learns a feature subspace for a group of tasks; it is introduced in [68], and specified in (2.9).

Two Settings. We use two settings for evaluation. First, we have access to a balanced set of labeled data from all viewpoints. We use the standard two-thirds and one-third split for training and testing, respectively. This setting allows us to compare our LMTL with the baselines and methods which use all viewpoints in training. The second setting tests how our LMTL deals with an unbalanced number of videos from different viewpoints, as is often the case in real

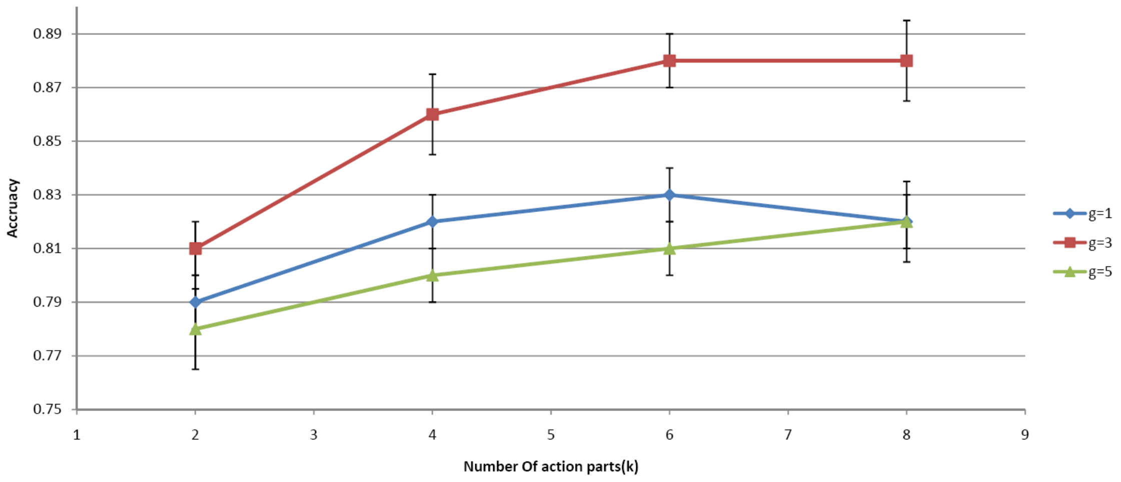


Figure 2.2: Our average recognition accuracy on i3DPost videos for different input parameters K and g.

world applications. We have access to videos from one or more source views, and limited (or no) access to videos from other target views. For evaluation, we vary the number of source views, and the number of videos from target views present in training.

Tables 2.1, 2.2 and 2.3 show the average accuracy of LMTL and the baselines with respect to different viewpoints on IXMAS, IXMAS(new) and i3DPost respectively in the first setting.

We see that sharing features across all viewpoints in Baseline 2 worsens results relative to Baseline 1. This is not a surprise, because the assumption that all viewpoints share a common feature space is too strong (e.g., top view in IXMAS dataset has completely different appearance from the other views). We can see that Baseline 3 gives better accuracy by grouping different viewpoints. Another interesting observation is the effect of using latent action parts. LMTL(g=1), gets better results compared to Baseline 3, especially on the IXMAS(new) and I3DPost datasets which contain occlusion and structured actions. This shows the merit of our accounting for action parts. We also see performance improvement by grouping viewpoints, LMTL(g=3), over using a single group. In summary, we perform $10\% - 26\%$ better than the baselines on the benchmark datasets.

Tables 2.4, 2.5 and 2.6 show the confusion tables of our approach for action classes on the IXMAS, IXMAS(new) and i3DPost datasets. Although we do not model structured actions and actions with more than one actor explicitly in our model, results on the i3DPost dataset show a

Target View	Cam0	Cam1	Cam2	Cam3	Cam4	Avg
B1	78.7	75.3	74.8	73.8	69.6	74.4
B2	78.9	71.5	70.1	69.1	72.4	72.4
B3(g=3)	81.1	82.8	82.5	80.4	77.6	80.9
LMTL(g=1)	86.4	85.5	80.2	84.1	76.8	82.6
LMTL(g=3)	96.8	95.6	94.7	96.5	92.1	95.1

Table 2.1: Average accuracy in [%] of LMTL and Baseline1 (B1), Baseline2 (B2), Baseline3 (B3) for different camera viewpoints on IXMAS.

Target View	Cam0	Cam1	Cam2	Cam3	Cam4	Avg
B1	65.5	64.2	62.7	68.8	59.3	64.1
B2	60.3	66.2	60.5	63.8	61.9	62.5
B3(g=3)	68.8	70.1	66.5	70.6	64.4	68.1
LMTL(g=1)	78.2	81.6	80.7	77.6	76.1	78.8
LMTL(g=3)	90.2	91.4	88.7	88.1	84.4	88.6

Table 2.2: Average accuracy in [%] of LMTL and Baseline1 (B1), Baseline2 (B2), Baseline3 (B3) for different camera viewpoints on IXMAS(new).

Target View	Cam0	Cam1	Cam2	Cam3	Cam4	Cam5	Cam6	Cam7	Avg
B1	72.4	74.8	72.6	69.6	69.7	70.6	73.9	71.2	71.9
B2	69.7	73.9	72.3	68.8	70.5	69.7	70.4	70.1	70.7
B3(g=3)	71.3	81.2	79.4	80.7	74.5	77.3	78.2	77.8	77.6
LMTL(g=1)	85.4	84.5	86.7	86.6	81.6	79.5	79.8	80.9	83.1
LMTL(g=3)	89.9	91.1	89.5	90.9	86.9	85.6	84.2	83.4	87.7

Table 2.3: Average accuracy in [%] of LMTL and Baseline1 (B1), Baseline2 (B2), Baseline3 (B3) for different camera viewpoints on i3DPost.

	CW	CA	GU	KI	PU	PT	PC	SH	SD	TA	WK	WV
CW	93.1	0	0	0	0	1.7	0	3.4	0	0	0	1.7
CA	3.4	89.7	0	0	0	0	0	1.7	0	0	0	5.2
GU	0	0	100	0	0	0	0	0	0	0	0	0
KI	0	0	0	100	0	0	0	0	0	0	0	0
PU	0	0	0	0	98.3	0	0	0	1.7	0	0	0
PT	0	0	0	0	0	86.2	8.6	0	1.7	0	0	3.4
PC	0	0	0	3.4	0	3.4	93.1	0	0	0	0	0
SH	1.7	1.7	0	0	0	0	0	89.7	0	0	0	6.9
SD	0	0	0	0	0	0	0	0	100	0	0	0
TA	0	0	0	0	1.7	0	0	0	0	96.6	1.7	0
WK	0	0	0	0	0	0	0	0	0	0	100	0
WV	1.7	3.4	0	0	0	1.7	0	1.7	0	0	0	91.4

Table 2.4: The confusion matrix of LTML for the IXMAS action classes. CW=CheckWatch, CA=CrossArms, GU=GetUp, KI=Kick, PU=PickUp, PT=Point, PC=Punch, SH=ScratchHead, SD=SitDown, TA=TurnAround, WK=Walk, WV=Wave. The values are in [%]

reasonable accuracy for these set of actions.

Studying recognition accuracy per viewpoint is important, because it shows how well an approach performs in different viewpoints. Fig. 2.3 shows our average accuracy per viewpoint on the IXMAS dataset. We can see that our recognition accuracy is consistent across different viewpoints.

Table 2.7 shows the confusion matrix of our viewpoint estimation on the IXMAS dataset. Our average viewpoint estimation accuracy is 82%. Confusion matrices of our viewpoint estimation on the IXMAS(new) and i3DPost datasets are shown in tables 2.8 and 2.9 respectively.

In the second setting, we fix the number of source viewpoints, and evaluate the sensitivity of LMTL to a varying fraction of target samples in training. Fig. 2.4 shows the average accuracy of LMTL for different fractions of target views in the IXMAS datasets. For 0% fraction of target views, LMTL does not perform as good as [91] on IXMAS. This is because LMTL is

	CW	CA	GU	KI	PU	PC	SH	SD	TA	WK	WV
CW	79.3	8.6	0	0	0	5.2	3.4	0	0	0	3.4
CA	6.9	75.9	0	0	0	3.4	6.9	0	0	0	6.9
GU	0	0	89.7	0	3.4	0	0	6.9	0	0	0
KI	0	0	0	93.1	0	5.2	0	0	1.7	0	0
PU	0	0	5.2	0	94.8	0	0	0	0	0	0
PC	1.7	0	0	6.9	0	91.4	0	0	0	0	0
SH	0	5.2	0	0	0	1.7	82.8	0	0	0	10.3
SD	0	0	6.9	0	3.4	0	0	89.7	0	0	0
TA	0	0	0	1.7	0	0	0	0	93.1	5.2	0
WK	0	0	0	0	0	0	0	0	0	100	0
WV	0	3.4	0	1.7	0	0	5.2	0	0	5.2	84.5

Table 2.5: Confusion matrix of LMTL on IXMAS(new) action classes. CW=CheckWatch, CA=CrossArms, GU=GetUp, KI=Kick, PU=PickUp, PC=Punch, SH=ScratchHead, SD=SitDown, TA=TurnAround, WK=Walk, WV=Wave

	BD	HS	HW	JF	JP	PL	RN	RF	RJ	SS	WK	WS
BD	96.6	0	0	0	0	0	0	0	0	3.4	0	0
HS	0	65.5	0	0	0	0	0	0	1.7	0	24.1	8.6
HW	0	0	100	0	0	0	0	0	0	0	0	0
JF	0	0	0	91.4	0	0	8.6	0	0	0	0	0
JP	0	0	0	0	100	0	0	0	0	0	0	0
PL	3.4	5.2	0	10.3	0	67.2	3.4	1.7	0	8.6	0	0
RN	0	0	0	0	0	0	96.6	1.7	0	0	1.7	0
RF	0	0	0	0	0	0	5.2	93.1	1.7	0	0	0
RJ	0	0	0	15.5	0	0	8.6	3.4	70.7	0	0	1.7
SS	6.9	0	0	0	0	0	0	0	0	93.1	0	0
WK	0	0	0	0	0	0	0	0	0	0	100	0
WS	0	0	0	0	0	0	1.7	0	0	0	20.7	77.6

Table 2.6: Confusion matrix of LMTL on I3DPost action classes. BD=Bend, HS=HandShake, HW=HandWave, JF=JumpForward, JP=Jump-In-Place, PL=Pull, RN=Run, RF=Run-Fall, RJ=Run-Jump-Walk, SS=Sit-Standup, WK=Walk, WS=Walk-Sit

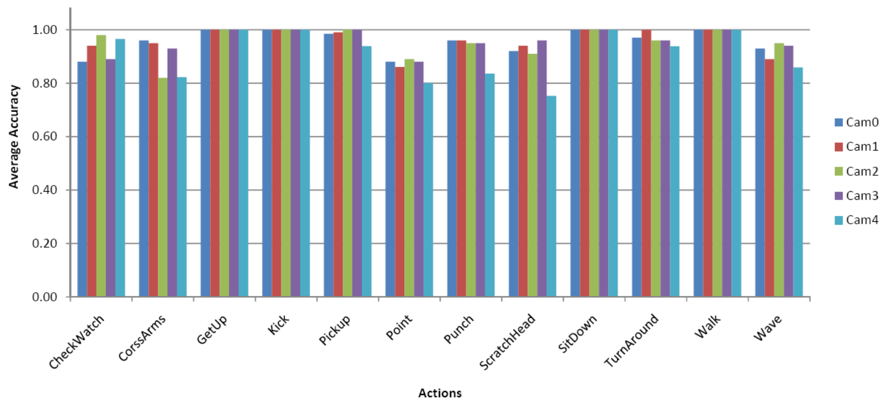


Figure 2.3: Accuracy in [%] of LMTL across different viewpoints on IXMAS.

	CAM0	CAM1	CAM2	CAM3	CAM4
CAM0	81.8	12.7	3	1.8	0.6
CAM1	16.4	81.2	1.8	0.6	0
CAM2	0.6	3	84.8	11.5	0
CAM3	3.6	3.6	8.5	83.6	0.6
CAM4	1.8	4.8	7.3	7.9	78.2

Table 2.7: The confusion matrix of viewpoints estimated by LMTL on IXMAS. The values are in [%]

	CAM0	CAM1	CAM2	CAM3	CAM4
CAM0	82.4	1.8	9.7	1.2	4.8
CAM1	1.2	87.9	2.4	7.9	0.6
CAM2	4.2	0.6	83.6	1.8	9.7
CAM3	1.2	8.5	1.8	86.1	2.4
CAM4	5.5	1.2	7.3	2.4	83.6

Table 2.8: Confusion matrix of estimated viewpoints for LMTL on IXMAS(new). Values are in [%]

	CAM0	CAM1	CAM2	CAM3	CAM4	CAM5	CAM6	CAM7
CAM0	88.7	6.7	4.6	0	0	0	0	0
CAM1	3.6	90.3	5.6	0	0	0	0.5	0
CAM2	0	0	90.3	0	1.0	8.7	0	0
CAM3	6.7	4.1	0	88.7	0	0.5	0	0
CAM4	0	0	0.5	0	86.7	0	2.1	10.8
CAM5	0	0	7.7	0	0	91.3	1.0	0
CAM6	0	0	0	0	3.6	0	91.8	4.6
CAM7	0	0.5	0	0	8.7	0	6.2	84.6

Table 2.9: Confusion matrix of estimated viewpoints for LMTL on I3DPost. Values are in [%]

a supervised approach, and needs at least some fraction of target examples for classification. Starting from one-fourth of target samples, we get better accuracy compared to [91].

In the second setting, we also test our sensitivity to the number of source viewpoints. This is important, because not all methods result in significant performance increase by using multiple source views (e.g. [91]). Fig. 2.5 shows the effect of using multiple source views on our average accuracy for three datasets. We have averaged over all different combinations of source views. In addition to the source view videos, we also use one-third of videos of the target viewpoint in learning. Note that the total number of groups of viewpoints is limited by the number of the source views, which is 5 in IXMAS, and 8 in both IXMAS(new) and i3DPost.

For a fair comparison with the state of the art, we use two different modes of tests: 1) Unbalanced labeled mode, where we use one-third of videos from the target views in training, and 2) Balanced labeled mode, where we use two-thirds of videos from the target views in training. The state-of-the-art approaches include [66] and [91, 99] as representatives of view-invariant features and transfer learning approaches. [91] uses multi-kernel SVM. We also compare with another latent structured model [174]. A comparison with geometric-based approaches to view-invariant recognition is not possible, because they do not report their accuracy per viewpoint. Table 2.10 shows the comparison on IXMAS in the unbalanced labeled mode. Tables 2.11 and 2.12 show the comparison using fully labeled training data on the two IXMAS datasets. LMTL outperforms the state-of-the-art approaches by 4.5–6%.

The best average accuracy on the simple action classes of i3DPost, reported in [60], is 90.88%. From Table 2.3, LMTL outperforms the approach of [60] by 5.4% for the same set of actions. Our evaluation on i3DPost shows that the DPM representation of action classes is

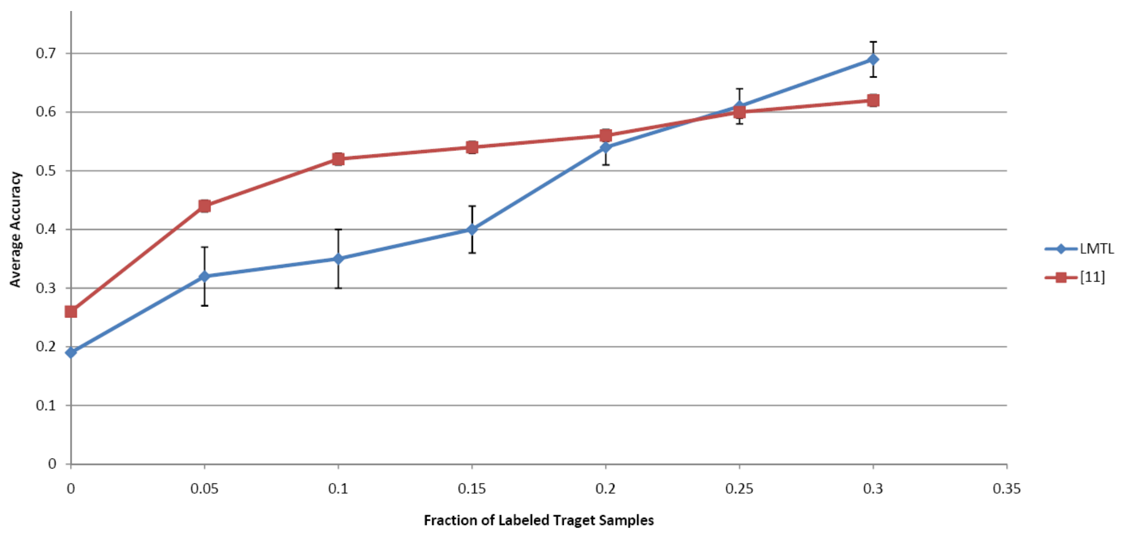


Figure 2.4: Average accuracy in [%] based on the fraction of target examples in the IXMAS training dataset.

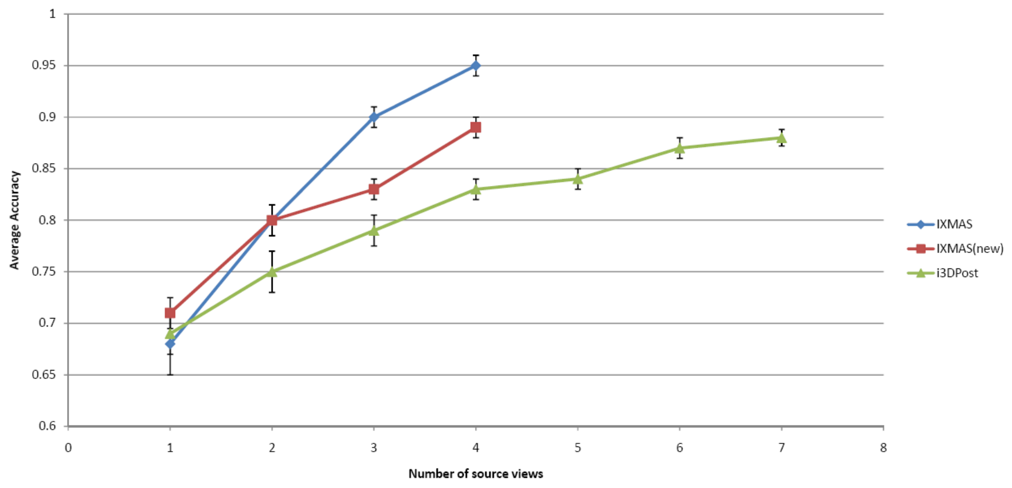


Figure 2.5: Average accuracy in [%] for different numbers of viewpoints in the IXMAS, IXMAS(new) and i3DPost datasets.

Target View	Cam0	Cam1	Cam2	Cam3	Cam4	Avg
[91]	62.0	65.5	64.5	69.5	57.9	63.9
[173]	86.1	93.1	73.6	80.6	-	83.3
LMTL	89.2	87.7	86.8	90.5	79.6	86.8
LMTL	89.2	87.7	86.8	90.5	-	88.6

Table 2.10: Average accuracy in [%] of LMTL, and the state of the art on IXMAS in unbalanced labeling mode.

Target View	Cam0	Cam1	Cam2	Cam3	Cam4	Avg
[66]	74.8	74.5	74.8	70.6	61.2	71.2
[99]	86.6	81.1	80.1	83.6	82.8	82.8
[173]	95.1	89.6	91.7	90.3	-	91.7
LMTL	95.9	96.8	94.5	96.9	89.9	94.8
LMTL	95.9	96.8	94.5	96.9	-	96

Table 2.11: Average accuracy in [%] of LMTL and the state of the art on IXMAS in balanced labeling mode.

Target View	Cam0	Cam1	Cam2	Cam3	Cam4	Avg
[160]	87.0	88.3	85.6	87.0	69.7	83.5
LMTL	90.2	91.4	88.7	88.1	84.4	88.6

Table 2.12: Average accuracy in [%] of LMTL and the state of the art on IXMAS(new) in balanced labeling mode.

capable of handling more complex, structured actions.

Implementation is done in C++. We perform our experiments on a core-i7 cpu and 8GB RAM PC. The inference running time is $O(m \log m)$, where m is the number of overlapping 2D+t windows in the video.

2.8 Summary

We have formulated a new approach to view-invariant action recognition. Our novelty is two-fold. We have formalized viewpoints of a given set of action classes as learning tasks, which can be jointly learned within the Multitask Learning (MTL) framework. To express that some viewpoints may not be correlated, and that discriminative action parts are subject to occlusion across the views, we have extended the standard MTL to latent MTL (LMTL). Thus, our LMTL identifies groupings of correlated viewpoints, leveraging a multiclass deformable parts model of actions.

Our evaluation on the benchmark IXMAS, IXMAS(new), and i3DPost datasets shows that accounting for parts and grouping viewpoints in LMTL leads to significant performance improvements over MTL, and other knowledge-transfer approaches to view-invariant action recognition.

Chapter 3: Regularizing Learning of Human Actions from Videos Using 3D Human-Skeleton Sequences

Abstract

This paper argues that large-scale action recognition in video can be greatly improved by providing an additional modality in training data – namely, 3D human-skeleton sequences – aimed at complementing poorly represented or missing features of human actions in the training videos. For recognition, we use Long Short Term Memory (LSTM) grounded via a deep Convolutional Neural Network (CNN) onto the video. Training of LSTM is regularized using the output of another encoder LSTM (eLSTM) grounded on 3D human-skeleton training data. For such regularized training of LSTM, we modify the standard backpropagation through time (BPTT) in order to address the well-known issues with gradient descent in constraint optimization. Our evaluation on three benchmark datasets – Sports-1M, HMDB-51, and UCF101 – shows accuracy improvements from 1.7% up to 14.8% relative to the state of the art.

3.1 Introduction

This paper is about classifying videos of human actions. We focus on domains that present challenges along two axes. First, we consider a large set of action classes (e.g., the Sports-1M dataset with 487 action classes [72]) which may have very subtle inter-class differences and large intra-class variations (e.g., Sports-1M contains 6 different types of bowling, 7 different types of American football and 23 types of billiards). The actions ma y be performed by individuals or groups of people (e.g., skateboarding vs. marathon), and may be defined by a particular object of interaction (e.g., bull-riding vs. horseback riding). Second, our videos are captured in uncontrolled environments, where the actions are viewed from various camera viewpoints and distances, and under partial occlusion.

Recent work uses Convolutional Neural Networks (CNNs) to address the above challenges [10, 63, 144, 72, 184, 25, 158, 134]. However, despite the ongoing research efforts to: (a) Increase the amount of training data [72], (b) Fuse hand-designed and deep-convolutional features

[158, 134, 184], and (c) Combine CNNs with either graphical models [144, 55, 112], or recurrent neural networks [25, 112] for capturing complex dynamics of human actions, we observe that their classification accuracy is still markedly below the counterpart performance on large-scale image classification. This motivates us to seek a novel deep architecture, leveraging some of the promising directions in (a)–(c).

Our key idea is to augment the training set of videos with additional data coming from another modality. This has the potential to facilitate capturing important features of human actions poorly represented in the training videos, or even provide complementary information missing in the videos. Specifically, in training, we use 3D human-skeleton sequences of *a few* human actions to regularize learning of our deep representation of all action classes. This regularization rests on our hypothesis that since videos and skeleton sequences are about human motions their respective feature representations should be similar. The skeleton sequences, being view-independent and devoid of background clutter, are expected to facilitate capturing important motion patterns of human-body joints in 3D space. This, in turn, is expected to regularize, and thus improve our deep learning from videos.

It is worth noting that the additional modality that we use in training *does not* provide examples of most action classes from our video domain. Rather, available 3D human-skeleton sequences form a small-size training dataset that is insufficient for robust deep learning. Nevertheless, in this paper, we show that accounting for this additional modality greatly improves our performance relative to the case when only videos are used in training.

As illustrated in Fig. 4.1, for action recognition, we use Long Short Term Memory (LSTM) grounded via a deep Convolutional Neural Network (DCNN) onto the video. LSTM is modified to have an additional representational layer at the top, aimed at extracting deep-learned feature $\boldsymbol{r}_v$ from the entire video. In training, we regularize learning of LSTM such that its output $\boldsymbol{r}_v$ is similar to features $\boldsymbol{r}_s$ produced by another encoder LSTM (eLSTM) grounded onto 3D human-skeleton sequences. The sequences record 3D locations of 18 joints of a human body while the person performs certain actions, such as those in the Carnegie-Mellon Mocap dataset [1] and the HDM05 Mocap dataset [110]. eLSTM is learned in an unsupervised manner by minimizing the data reconstruction error. Note that a hypothetical supervised learning of an LSTM on skeleton data would not be possible in our setting, since we have access to a small dataset representing only a small fraction (or none) of action classes from the video domain. During test time, we do not use any detections of human joints and their trajectories, but classify a new video only based on raw pixels taken as input to the LSTM+DCNN.

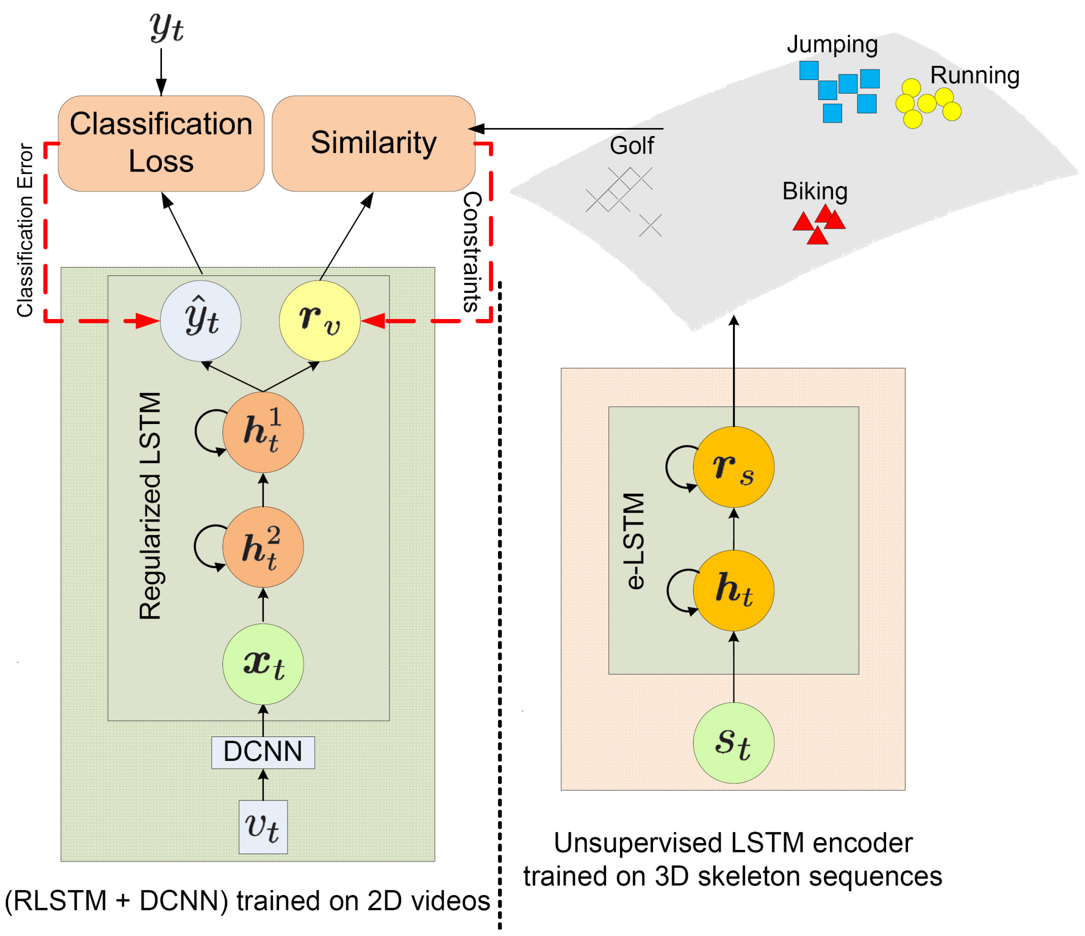


Figure 3.1: Our novel deep architecture: the LSTM on the left is trained on videos under weak supervision, and the encoder LSTM (eLSTM) on the right is trained in unsupervised manner on 3D human-skeleton sequences. v_t and s_t denote the input video frame and skeleton data at time t. $\boldsymbol{r}_v$ and $\boldsymbol{r}_s$ are the output features of LSTM and eLSTM. y and $\hat{y}$ are the ground-truth and predicted class labels. $\boldsymbol{h}_t$, $\boldsymbol{h}_t^1$, and $\boldsymbol{h}_t^2$ are the hidden layers in the two respective LSTMs. $\boldsymbol{x}_t$ is the output feature of DCNN's FC_7 layer. Euclidean distances between corresponding features $\boldsymbol{r}_v$ and $\boldsymbol{r}_s$ are jointly used with the prediction loss between y and $\hat{y}$ for a regularized learning of the LSTM on videos.

Our main contribution represents a novel regularization of LSTM learning. Unlike the standard regularization techniques, such as drop out or weight decay [58, 25, 183], we define a set of constraints aimed at reducing the Euclidean distances between top-layer features of LSTM trained on videos and corresponding output features of eLSTM. We use these constraints to regularize and thus extend the standard backpropagation through time (BPTT) algorithm [25]. BPTT back-propagates a class-prediction loss for updating LSTM parameters via stochastic gradient descent. We additionally back-propagate the above constraints between corresponding features. This requires modifying the standard (unconstrained) gradient descent to an algorithm that accounts for constraints. To this end, we use the hybrid steepest descent [41].

In this paper, we consider several formulations of regularizing LSTM learning corresponding to the cases when ground-truth class labels are available for the skeleton sequences, and when this ground truth is not available.

We present experimental evaluation on three benchmark datasets, including Sports-1M [72], HMDB-51 [81], and UCF101 [137]. We report the performance improvement ranging from 1.7% to 14.8% relative to the state of the art.

In the following, Sec. 3.2 reviews related work, Sec. 3.3 specifies LSTM, Sec. 3.4 formulates our novel deep architecture, and Sec. 3.5 presents our results.

3.2 Closely Related Work

This section reviews related work on: combining CNN and LSTM, encoder LSTM, using skeleton data for action classification, and multimodal deep learning in vision.

- **LSTM+DCNN.** Frame-level DCNN features have been used as input to LSTM for action classification [25]. This architecture has been extended with additional layers for convolutional temporal pooling [112]. The main advantages include that these architectures are deeply compositional in both space and time, and that they are capable of directly handling variable-length inputs (e.g., video frames) and capturing long-range, complex temporal dynamics. They are learned with backpropagation through time (BPTT). Our key difference is in the definition of LSTM output layer. Our output layer includes the standard softmax layer for classification and the additional representational layer for mapping input video sequences to a feature space. It is the construction of this new feature space that allows us to regularize LSTM learning. Another difference is that we replace the standard stochastic gradient descent in BPTT, with an algorithm that accounts for constraints in optimization.

- **Encoder LSTM.** LSTM has been used to encode an input data sequence to a fixed length vector, which, in turn, is decoded to predict the next unobserved data [138]. However, this recurrent autoencoder-decoder paradigm has not yet demonstrated competitive performance on action classification in videos. Our main difference is that we use the encoder LSTM to generate a feature manifold for regularizing a supervised learning of another LSTM.

- **Action classification using skeleton data** has a long-track record [47, 103, 150, 157, 171, 175, 26]. However, at test time, these approaches require either 3D locations of human joints, or detection of human joints in videos. In contrast, at test time, we just use pixels as input, and neither detect human joints nor need their 3D locations.

- **Multimodal learning.** Recent work uses text data as an additional modality to improve image classification [114, 38, 28, 135, 78, 105]. For example, a combination of DCNN and LSTM has been used for multimodal mapping of finer object-level concepts in images to phrases [78].

Closely related work introduces a semi-supervised embedding in deep architectures [165].

Due to the fundamental differences in our problem statements, we cannot use these approaches. More importantly, instead of a heuristic combination of classification loss and the embedding loss, we use a well-defined set of constraints to explicitly exploit the information contained in the embedding space.

Another difference is that object classes of text data are in one-to-one correspondence with object classes appearing in images (or it is assumed that the image and text domains have a large overlap of object classes). In contrast, our 3D skeleton sequences represent only a few action classes from the video domain. Similar to [38, 135], we do not use the other modality at test time. For multimodal training, these approaches use, for example, the Euclidean distance [135], modified hinge loss [38], parameter transfer and regression loss [28], or pairwise ranking loss [78]. Instead, we minimize the cross entropy loss associated with the softmax layer of LSTM, subject to the feature similarity constraints in our regularization. Importantly, in Sec. 3.4.3, we provide convergence guarantees for our regularized learning of LSTM.

3.3 A Brief Review of LSTM

A major building block of our novel architecture is LSTM [58, 138], depicted in Fig. 3.2. LSTM is a recurrent neural network as briefly reviewed below.

The LSTM's recurrent unit memorizes previous input information in a memory cell, c_t, indexed by time t. The output is hidden variable h_t. Three gates control the updates of c_t and h_t: 'input' i_t, 'output' o_t, and 'forget' f_t. In (3.1), we summarize their update equations, where symbol $\odot$ denotes the element-wise product, and W are LSTM parameters.

$$\begin{aligned}
i_t =& \sigma(W_{xi}\boldsymbol{x}_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i),\\
o_t =& \sigma(W_{xo}\boldsymbol{x}_t + W_{ho}h_{t-1} + W_{co}c_{t-1} + b_o),\\
f_t =& \sigma(W_{xf}\boldsymbol{x}_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f),\\
c_t =& f_t \odot c_{t-1} + i_t \odot \tanh(W_{xc}\boldsymbol{x}_t + W_{hc}h_{t-1} + b_c),\\
h_t =& o_t \odot \tanh(c_t).
\end{aligned} \tag{3.1}$$

From (3.1), the input gate, i_t, regulates the updates of c_t, based on inputs $\boldsymbol{x}_t$ and previous values of h and c. The 'output gate', o_t, controls if h_t should be updated given c_t. The forget gate, f_t, resets the memory to its initial value. The LSTM parameters $W_{\text{LSTM}} = \{W_{xi}, W_{xo}, W_{xf}, W_{ci}, W_{co}, W_{cf}, W_{hi}, W_{ho}, W_{hf}\}$ are jointly learned using BPTT.

The LSTM unit preserves error derivatives of the deep unfolded network. This has been shown to avoid the well-known vanishing gradient problem, and allows LSTM to capture long-range dependencies in the input data sequence.

3.4 Regularizing LSTM for Action Recognition

As mentioned in Sec. 3.1, our novel architecture consists of eLSTM for learning a feature representation of 3D human-skeleton sequences, and a stacked DCNN+LSTM for classifying videos. Below, we describe these two components.

eLSTM. Fig. 3.3 shows the time-unfolded encoder and decoder parts of eLSTM. The goal of eLSTM is to encode the input skeleton sequence, $s = \{s_t : t = 1, 2, \dots\}$, consisting of 3D locations of human joints. The sequences may have variable lengths in time. eLSTM observes the entire skeleton sequence s, and encodes it to a feature vector $\boldsymbol{r}_s$. The set of encoded representations $\{\boldsymbol{r}_s\}$ are assumed to form a manifold $\mathcal{M}_s$ of skeleton sequences. To learn the encoder, a decoder LSTM tries to reconstruct the normalized input 3D data in the reverse order.

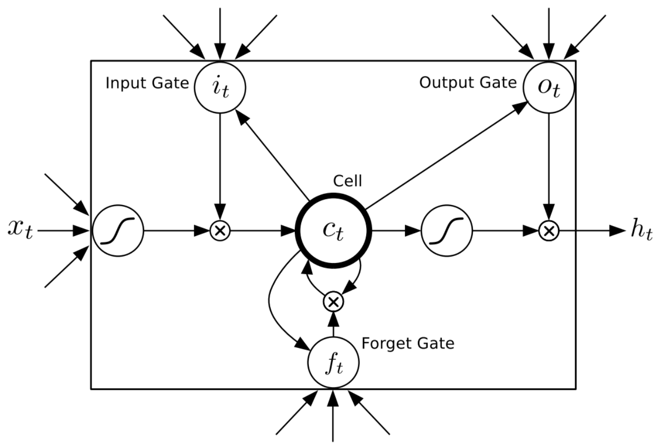


Figure 3.2: LSTM unit [58, 138].

The reconstruction error is then estimated in terms of the mean-squared error in the normalized 3D coordinates, and used to jointly learn both the encoder and decoder LSTMs. The reversed output reconstruction benefits from low range correlations which makes the optimization problem easier. The input to the encoder at each time step t, is the output of the decoder at time step $t-1$, i.e. s_{t-1}. An alternative architecture is to learn an encoder LSTM to predict the next skeleton frame. To avoid over-fitting, we use the standard drop-out regularization for eLSTM [183].

DCNN+LSTM. As shown in Fig. 4.1, for classifying human actions in videos we use a stacked architecture of a frame-level DCNN and a two-layer LSTM. DCNNs have been demonstrated as capable of learning to extract good image descriptors for classification [147, 184, 71]. For DCNN, we use the same network architecture initially trained on the ImageNet dataset as in [142]. The only difference is in the number of output units of the softmax layer at the top of DCNN, since we address different numbers of action classes. Note that we later fine-tune DCNN parameters together with LSTM ones in our regularized learning.

Our LSTM differs from the model used in [25] in the top output layer. The output layer of our LSTM extends the standard fully connected softmax layer for classification with an additional representation layer. This representation layer is aimed at mapping input video sequences, $v = \{v_t : t = 1, 2, \dots\}$, with variable lengths in time, to fixed-length vectors $\boldsymbol{r}_v$. The size of this representation layer is set to be equal to that of the output layer of eLSTM. Thus, the vectors $\boldsymbol{r}_v$ and $\boldsymbol{r}_s$ have the same size.

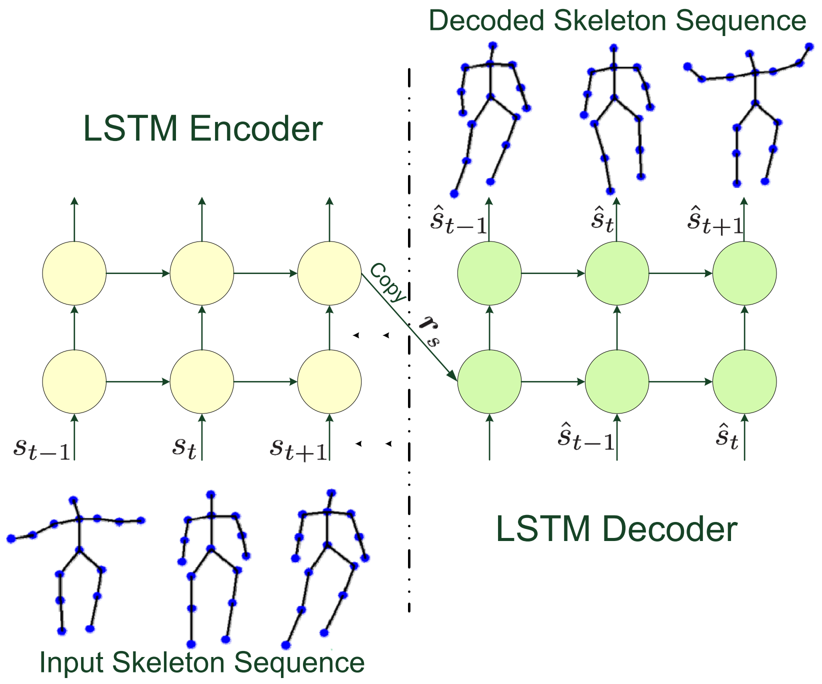


Figure 3.3: The time-unfolded visualization of the encoder LSTM (left) and decoder LSTM (right) for learning a feature representation from input 3D human-skeleton sequences. The encoder LSTM observes the entire skeleton sequence and encodes it to a fixed-length representation. The decoder LSTM tries to reconstruct 3D locations of human joints in the reverse order of the input sequence. The reconstruction error is then used to jointly learn both the encoder and decoder LSTMs.

Our goal of learning DCNN+LSTM parameters, Θ, is to minimize the classification loss, $L(\Theta)$, subject to constraints g between vectors $\boldsymbol{r}_v$ and corresponding vectors $\boldsymbol{r}_s$ in manifold $\mathcal{M}_s$. Thus, for all training videos $v \in D_v$, we formulate the regularized learning of DCNN+LSTM as

$$\begin{aligned} &\min_{\Theta} L(\Theta) \\ &\text{s.t. } \forall v \in D_v \ \ , g(\boldsymbol{r}_v, \mathcal{M}_s) \leq 0, \end{aligned} \tag{3.2}$$

where $L(\Theta) = \sum_{v \in D_v} l(v, \Theta)$ is the cross entropy loss associated with the softmax layer of LSTM, and g is a constraint based on the distance between $\boldsymbol{r}_v$ and $\mathcal{M}_s$. g can be defined in different ways. We only require that the constraints are differentiable functions. In the following, we define two distinct constraints that give the best results in our experiments.

3.4.1 Class Independent Regularization

For class independent regularization of learning DCNN+LSTM parameters, the constraint $g = g_1$ in (3.2) is specified to ensure that, for every video $v \in D_v$, $\boldsymbol{r}_v$ is sufficiently similar to the mapped vectors $\boldsymbol{r}_s \in \mathcal{M}_s$. This is achieved by defining an upper-bound for the average distance between $\boldsymbol{r}_v$ and all $\boldsymbol{r}_s \in \mathcal{M}_s$ as in (3.3)

$$g_1(\boldsymbol{r}_v, \mathcal{M}_s) = \frac{1}{n} \sum_{\boldsymbol{r}_s \in \mathcal{M}_s} \|\boldsymbol{r}_v - \boldsymbol{r}_s\|_2^2 - \alpha, \tag{3.3}$$

where $\alpha > 0$ is an input parameter, and n is the number of training skeleton sequences. This type of regularization is suitable for cases when training skeleton sequences do not represent the same action classes as training videos or represent a very small subset of action classes.

3.4.2 Class Specific Regularization

For class specific regularization of learning DCNN+LSTM parameters, the constraint $g = g_2$ in (3.2) is specified to take into account action class labels of training skeleton sequences, when available. This type of learning regularization is suitable for cases when some action classes represented by training skeleton sequences do “overlap” with certain action classes in training videos. The definition of class equivalence between the two modalities can be easily provided along with ground-truth annotations.

We expect that $\boldsymbol{r}_v$ and $\boldsymbol{r}_s$ should be more similar if video v and skeleton sequence s share the same class label, $l_v = l_s$, than $\boldsymbol{r}_v$ and $\boldsymbol{r}_{s'}$ of skeleton sequences s' with different class labels, $l_v \neq l_{s'}$. This is defined in (3.4)

$$g_2(\boldsymbol{r}_v, \mathcal{M}_s) = \frac{1}{n_=} \sum_{\substack{\boldsymbol{r}_s \in \mathcal{M}_s \\ l_v = l_s}} \|\boldsymbol{r}_v - \boldsymbol{r}_s\|_2^2 - \frac{1}{n_{\neq}} \sum_{\substack{\boldsymbol{r}_{s'} \in \mathcal{M}_s \\ l_v \neq l_{s'}}} \|\boldsymbol{r}_v - \boldsymbol{r}_{s'}\|_2^2. \tag{3.4}$$

where $n_=$ and $n_{\neq}$ are the numbers of training skeleton sequences that have $l_v = l_s$ and $l_v \neq l_s$. The constraint g_2 defined in (3.4) ensures that the average Euclidean distance between $\boldsymbol{r}_v$ and $\boldsymbol{r}_s \in \mathcal{M}_s$ for skeleton sequences s of the same action label is less than the average Euclidean distance between $\boldsymbol{r}_v$ and $\boldsymbol{r}_{s'}$ for skeleton sequences s' of different action labels.

3.4.3 Hybrid Steepest Descent

We jointly learn parameters, $\Theta{=}W_{\text{LSTM}}{\cup}W_{\text{DCNN}}$, by modifying the standard backpropagation and applying a stochastic gradient descent algorithm which accounts for the aforementioned constraints. One standard approach to solve a constrained convex optimization problem is to use Lagrange multipliers and fuse the constraints with the original objective into a new unconstrained optimization problem. Unfortunately, as demonstrated in [117], gradient descent poorly works with Lagrange multipliers, mainly because they introduce saddle points in the new objective. Therefore, we resort to an alternative approach called hybrid steepest descent [41] for solving our constrained convex optimization, as described below.

Fig. 3.4 shows a simulation of hybrid steepest descent for a simple convex optimization problem. At each iteration, the algorithm checks if the current solution – i.e., the current Θ parameters in our case – satisfies all constraints. If so, Θ is updated according to the gradient of the original objective function – i.e., in our case, the cross entropy loss $L(\Theta)$ – without considering the constraints. If any constraint is violated by the current solution – i.e in our case, $g(\boldsymbol{r}_v, \mathcal{M}_s) \leq 0$ constraint is not satisfied for $\boldsymbol{r}_v$ computed by DCNN+LSTM with the current Θ parameters –, we update Θ according to the gradient of the violated constraints. The proof of correctness, asymptotic stability, and convergence of this algorithm is presented in [41]. Note that there is no guarantee that the final value of the parameters Θ satisfies all constraints. In our implementation we keep track of the top 5 best solutions based which have the minimum number of violated constraints.

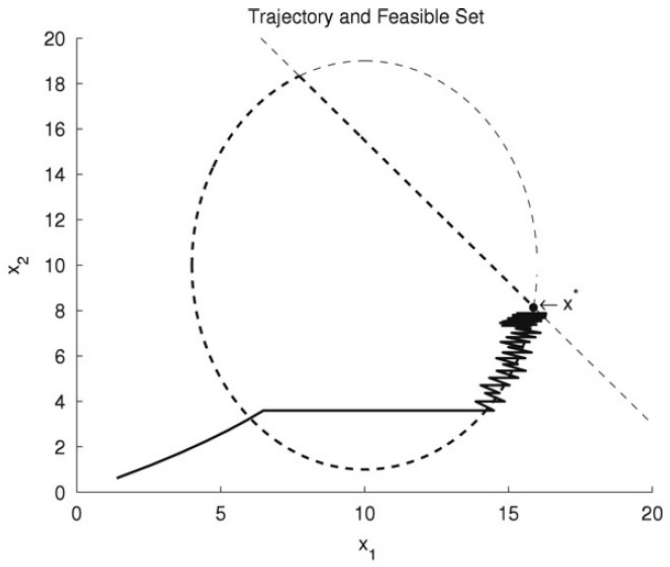


Figure 3.4: Simulation of the hybrid steepest descent algorithm for a simple optimization problem presented in [41]. In this example $\Theta = x_1, x_2$, $L(\Theta) = -x_1$, $g_1 = \frac{(x_1-10)^2}{36} + \frac{(x_2-10)^2}{81} - 1$, and $g_2 = \frac{10}{8}x_1 + x_2 - 28$. The dark hashed lines show the boundary of the feasible set.

We use hybrid steepest descent to modify BPTT for regularized learning of DCNN+LSTM. Note that the updates of the recurrent units only depend on the particular value of the the back-propagated gradient. Once this gradient is computed, we use it in the same way as in the standard BPTT. Thus, as summarized in Alg. 1, our modification of BPTT amounts to alternating the specification of the error gradient at the output layer according to the above rules of hybrid steepest descent. In Alg. 1, we use $g > 0$ to denote a constraint violation, and η_t is the time dependent learning rate.

3.5 Results

For evaluation, we use the Sports-1M [72], HMDB-51 [81], and UCF-101 [137] datasets. Sports-1M consists of more than 1 million videos from Youtube, annotated with 487 action classes. There are on average 3000 videos for each action class, where the average video length is 5 minutes 36 seconds. Considering the large number of action classes, long duration of the videos, and large variety of camera motions, Sports-1M is currently acknowledged as one of the most

Algorithm 1 Regularized learning procedure of LSTM

```
1:  Input: Training videos $D_v$ and $\mathcal{M}_s$
2:  Output:LSTM parameters $\Theta$
3:  % note that during training we assume sequences in $D_v$ are of the same length
4:  repeat
5:      for every $v \in D_v$ do
6:          forward-pass $v$ through unfolded LSTM
7:          if all constraints satisfied then
8:              $\Theta_t \xleftarrow{BPTT} \Theta_{t-1} - \eta_t \nabla L(\Theta)$
9:
10:             % This updates $\Theta$ by back-propagating the gradient of the cross entropy loss.
11:         else
12:             $\Theta_t \xleftarrow{BPTT} \Theta_{t-1} - \eta_t \sum_{g>0} \nabla g(\Theta);$
13:             % This updates $\Theta$ by back-propagating the sum of gradients of the violated
14:             constraints.
15:         end if
16:     end for
17: until
```

challenging benchmarks for action recognition in the wild. HMDB-51 consists of 6849 videos with 51 action labels. Each action class has at least 100 videos with an average video length of 3.1 seconds. UCF-101 consists of 13,320 videos of 101 action classes with average video length of 7.2 seconds. We follow [112], and report our average accuracy on the given three dataset splits.

We use human skeleton sequences of the Carnegie-Mellon Mocap [1] and HDM05 [110] datasets to train eLSTM. HDM05 consists of 2337 sequences with 130 action classes performed by 5 subjects. These sequences record the 3D locations of 31 human joints at each frame. Carnegie-Mellon Mocap consists of 2605 sequences with 23 action classes. These sequences record the 3D locations of 41 human joints at each frame. For consistency, we use only 18 human-body joints (head, lower back, upper back, upper neck, right/left clavicle, hand, humerus, radius, femur, tibia, foot) of the skeleton sequences, and resolve name conflicts and duplicates in these two datasets.

eLSTM: An LSTM with two hidden layers is used for the encoder and decoder. Input and output of the encoder and decoder LSTMs are $54 = 18 \times 3$ dimensional vectors of 3D human body joint positions. We normalize input data in the range of [0, 1]. We empirically verified that eLSTM with 512 and 1024 hidden units in the first and second recurrent layer of the encoder LSTM, and the same number of hidden units in a reverse order for the decoder LSTM results in the smallest reconstruction error. The model is trained on 16 frame sequences. Since the training is unsupervised, both Carnegie-Mellon Mocap and HDM05 datasets are used in training phase. It takes 16-19 hours to converge for about 3160 minutes of skeleton sequences. For defining g_2 constraints, we use the class labels defined in HDM05.

DCNN: We use GoogLeNet [142] trained on ImageNet [130] as DCNN in our approach. This DCNN is fine-tuned on a set of randomly sampled video frames. The output layer is modified for the fine-tuning based on the number of action classes. On average (150-200) frames are sampled from each video. The second to the last fully connected layer (FC_7) is used as the frame descriptor input to LSTM. Note that, later, DCNN parameters are fine-tuned again jointly with LSTM training.

Our Regularized LSTM (RLSTM): Our RLSTM for action recognition contains two hidden layers of 2048 and 1024 hidden units, respectively. The number of output units for classification is 487 for Sports-1M, 51 for HMDB-51, and 101 for UCF101. The number of output units for representation is 512, which is equal to the number of hidden units in the second recurrent layer of eLSTM. Similar specifications are used in [25, 138]. Fixed length sequences of 16 frames are used in training. We find that a random initialization of RLSTM converges to a poor local optimum. To overcome this, we train a non regularized LSTM using one-tenth of the training instances in Sports-1M. We use weights of this learned model to initialize weights of RLSTM. Weights of the representation layer are initialized randomly between [-0.1, 0.1]. Similar to [25, 138], we estimate an average prediction of 16 block frames with a stride of 8 in inference. The linear weighting presented in [112] is used to combine predictions from each block for the entire video.

Implementation: We use Caffe [64] and a modified version of the RNN library in Torch [19] in our experiments. All experiments are performed on an Intel quad core-i7 CPU and 16GB RAM PC with two Tesla-K80 Nvidia cards.

3.5.1 Baselines and Variations of our Approach

We conduct a comparison with several baselines in order to evaluate effectiveness of different constraints in our regularized learning of RLSTM. These baselines include the following:

1- **DCNN**: This is a 'single-frame' based action recognition evaluated in [72] – a video is represented by a single frame and classification is perform only based on the content in that frame

2- **LSTM**: This is a (DCNN+LSTM) learned without any regularization and constraints, similar to the approach of [25] with only difference in the number of hidden units (due to a different number of classes considered).

Based on the constraints, defined in Sec. 3.4.1 and 3.4.2, for regularizing learning of RLSTM, we define the following three variations of our approach:

1- **RLSTM**-$g1$: uses class independent constraints $g1$ to regularize the learning

2- **RLSTM-**$g2$: uses class specific constraints $g2$ to regularize the learning

3- **RLSTM-**$g3$: uses $g1 \cup g2$ to regularize the learning.

Table 3.1-a shows our average classification accuracy on HMDB-51 and UCF101. All variations of our method improved the accuracy of the baseline LSTM (3.1% to 12.2%). RLSTM-$g2$ achieves a better accuracy compared to RLSTM-$g1$. This strongly supports the hypothesis that deep-learned features of vides and skeleton sequences should be similar, and that our regularization should improve action recognition. Because the 3D human skeleton datasets and UCF101/HMDB-51 share a few common action classes, RLSTM-$g3$ which combines the class independent and class specific constraints outperforms RLSTM-$g2$.

Comparison of our method with the state of the art deep learning based approches is presented in Tab. 3.1-b. We can see that RLSTM-$g3$ outperforms variations of [72, 138, 25, 147, 134] which only use raw frames by $1.7\% - 22\%$. Higher accuracy is reported in [184, 112, 134, 158] on UCF101 for (raw pixels + optical flow) input. In comparison with their accuracy of $88.6\% - 90.3\%$, we achieve a comparable performance of 86.9% accuracy on UCF101 *by using only pixels of video frames*.

Hit@k values are a standard evaluation for large-scale datasets. A test instance is considered correctly labeled if the ground truth label is among the top k predictions. Similar to [112, 71] we use Hit@1 and Hit@5 values to report accuracy on Sports-1M. Table 3.2 shows the classification accuracy of the baselines, different RLSTM models, and the state of the art on Sports-1M. One interesting result is that RLSTM-g_1 outperforms RLSTM-g_2. We believe that this is because of

Method	UCF101	HMDB-51
Single-Frame [72]	64.9	41.2
LSTM	75.2	43.1
RLSTM-$g1$	78.3	49.3
RLSTM-$g2$	81.5	51.4
RLSTM-$g3$	86.9	55.3

(a)

Method	UCF101	HMDB-51
[72]	65.4	-
[138]	75.8	44.1
[25]	71.12	-
[134]	72.8	40.5
[184]	79.34	-
[147]	85.2	-
RLSTM-$g3$	86.9	55.3

(b)

Table 3.1: (a) Average classification accuracy of regularized LSTM models and baselines on UCF101 and HMDB-51. Our approach outperform the LSTM baseline by $11.7\% - 12.2\%$ (b) Average classification accuracy of RLSTM-$g3$ and the state of the art on UCF101 and HMDB-51. Our approach outperform the best result in state of the art by $1.7 - 11.2\%$.

a poor overlap of the large number of action classes in Sports-1M with the set of action classes in the skeleton data obtained from HDM05 and CMU Mocap. Our best approach improves the classification accuracy on Hit@1 by $2.5\% - 16.6\%$. Also the baseline non-regularized LSTM yields better accuracy than the temporal pooling approaches of [72, 147]. We believe that this is mainly because the baseline LSTM has access to the longer video frames.

To verify the effectiveness of hybrid-gradient descent used in our regularized learning, we also train our DCNN+LSTM using AdaGrad [27] and Adam [75]. These two alternative algorithms are aimed at minimizing the standard weighted sum of the classification and representation loss. The comparison is shown in Table 3.3, where RLSTM denotes our approach to regularized training with hybrid-gradient descent, and LSTM($\cdot$) denotes our approach to regularized training with AdaGrad or Adam.

The regularized learning of RLSTM-$g1$ and RLSTM-$g3$ is controlled by the α parameter, specified in eq. (3.3). Fig. 3.5 shows how our accuracy changes for different values of α on HMDB-51. We can see that for small values of α the accuracy is very low. We observe that for these values the learning algorithm does not converge. The algorithm alternates gradient updates with respect to classification error and violated constraints. This is because the optimization problem becomes almost infeasible for small values of α. The accuracy gradually increases for larger values of α, but again decreases when α becomes sufficiently large. The accuracy of

Method	Hit@1	Hit@5
Single-Frame [72]	59.3	77.7
LSTM	71.3	89.9
[72]	60.9	80.2
[112]	72.1	90.6
[147]	61.1	85.2
RLSTM-$g1$	73.4	91.3
RLSTM-$g2$	62.2	85.3
RLSTM-$g3$	75.9	91.7

Table 3.2: Average classification accuracy of regularized LSTM models, baselines, and the state of the art on Sports-1M. Our approach outperform the best result in state of the art by 2.5% and the LSTM baseline by 4.6%.

Method	UCF101	HMDB-51
LSTM(adm)-$g1$	75.7	44.6
LSTM(adm)-$g2$	79.20	49.8
LSTM(adm)-$g3$	81.8	51.3
LSTM(adg)-$g1$	74.9	44.3
LSTM(adg)-$g2$	81.6	49.5
LSTM(adg)-$g3$	80.7	48.6
RLSTM-$g1$	78.3	49.3
RLSTM-$g2$	81.5	51.4
RLSTM-$g3$	86.9	55.3

Table 3.3: Average classification accuracy of our approach on UCF101 and HMDB-51, when the regularized training is conducted with AdaGrad [27] or Adam [75] – denoted as LSTM(adg) and LSTM(adm) – or hybrid-gradient descent – denoted as RLSTM.

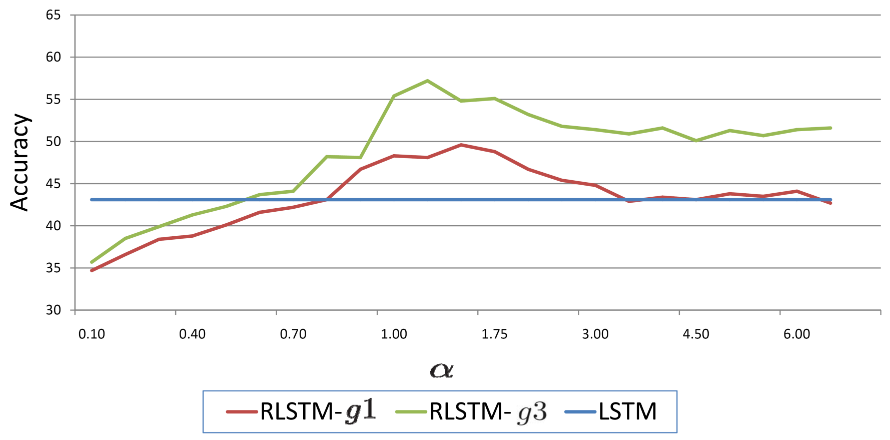


Figure 3.5: Average classification accuracy of RLSTM-$g1$ and RLSTM-$g3$ on HMDB-51 for different α values, where α is the input parameter that controls the regularized learning of RLSTM-$g1$ and RLSTM-$g3$. Small values of α enforce stronger regularization that the feature outputs of RLSTM and eLSTM are highly similar.

RLSTM-$g1$ reaches that of the standard LSTM for large values of α, and remains nearly the same. This is because sufficiently large values of α yield an unconstrained optimization, i.e., non-regularized learning of LSTM.

3.5.2 Contributions of 3D Skeleton Sequences

We conduct further experiments to understand the differences between the original and the regularized video model that have access to 3D skeleton information. We first compute the per-class average precision for all classes and highlight the ones that show largest differences. This is shown in table 3.4. The interesting observation is that although the overall average accuracy on the entire dataset is improved, this is not the case for all classes. We identified three different patterns: 1) For the action classes with instances in 3D skeleton sequences we see the highest accuracy improvement. 2) For the action classes which do not have corresponding instances in 3D skeleton sequences, but contain significant human motion we still observe a fair amount of improvement. 3) For action classes which do not have instances in 3D skeleton sequences and mostly involve motions not related to human motion (e.g. windsurfing) , we actually notice a small accuracy drop.

The other important question is that how much of the accuracy improvement is just a result

Actions	Accuracy Change
Running	+4.2
Badminton	+2.4
Track cycling	+2.1
Road bicycle racing	+2.4
Down hill biking	+1.9
bmx	+1.8
Wind Surfing	-1.2
Fishing	-1.0
Land Surfing	-0.0

Table 3.4: Average per-class accuracy improvement or drop on Sports-1M

Training Setup	Hit@1	Hit@5
100% video training data	71.3	89.9
99.5% video training data	71.2	89.9
99.5% video training data + 0.5% 3D sequence data	75.9	91.7

Table 3.5: Accuracy improvement based on amount of the additional data on Sports-1M

of an additional amount of data we use in training. In other words, we want to verify that the accuracy improvement is not just because of the extra training data but the type of data we use. This is important because eventually we are adding more training data and it is shown that more training data helps. To verify this we perform an experiment with fewer video data. The number of 3D skeleton sequences is only 0.5% of the available video data in training. We randomly choose 95.5% of the video data and compute the accuracy of a non-regularized video model. Later we train our regularized video model with the additional 0.5% 3D skeleton sequences. The result is shown in table 3.5. We can see that the addition of 3D skeleton sequence actually improves the accuracy 3% more than the same amount of video data.

3.6 Summary

We have proposed a novel deep architecture for large-scale action recognition. The main contribution of our work is to use 3D human-skeleton sequences to regularize the learning of LSTM, which is grounded via DCNN onto the video for action recognition. We have modified the back-propagation through time algorithm in order to account for constraints in our regularized joint learning of LSTM+DCNN. Our experimental results demonstrate that the skeleton sequences could successfully constrain the learning of LSTM+DCNN leading to an improved performance relative to the case when LSTM is trained only on videos. Specifically, on Sports-1M, UCF101, and HMDB-51 our accuracy improves by $2.5\% - 16.6\%$, $1.7 - 21.5\%$, and $11.2 - 14.8\%$, respectively. We also verified that the accuracy improvement is solely because of the type of the additional input data and not just the amount of the added data.

Chapter 4: Video Summarization through Generative Adversarial Networks

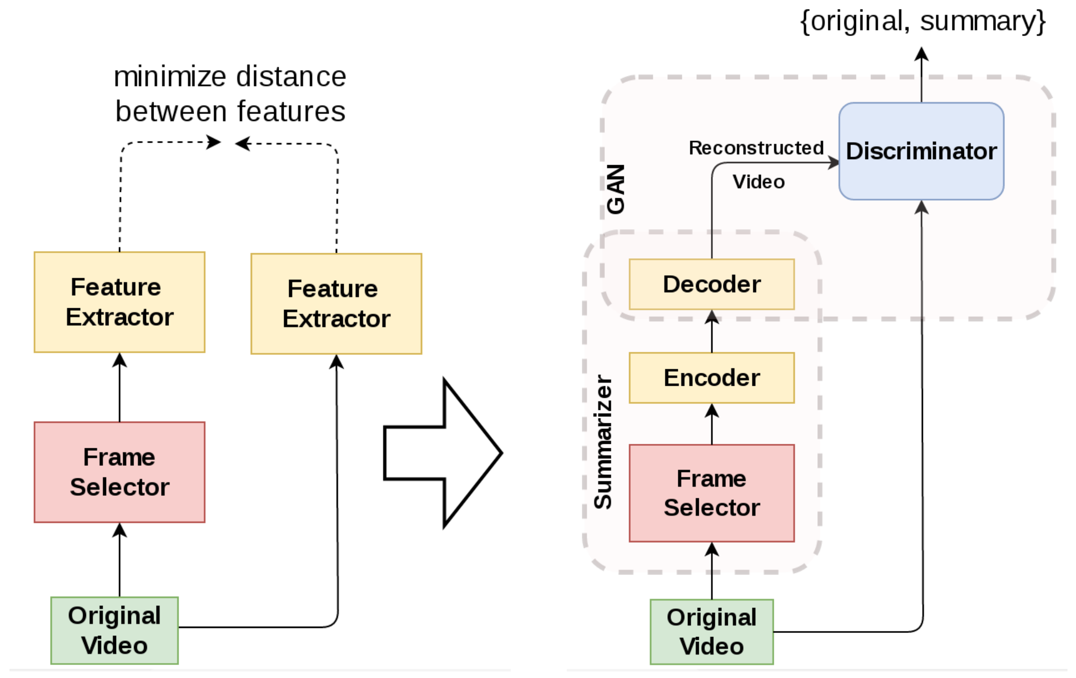
Abstract

This paper addresses the problem of unsupervised video summarization, formulated as selecting a sparse subset of video frames that optimally represent the input video. Our key idea is to learn a deep summarizer network to minimize distance between training videos and a distribution of their summarizations, in an unsupervised way. Such a summarizer can then be applied on a new video for estimating its optimal summarization. For learning, we specify a novel generative adversarial framework, consisting of the summarizer and discriminator. The summarizer is the autoencoder long short-term memory network (LSTM) aimed at, first, selecting video frames, and then decoding the obtained summarization for reconstructing the input video. The discriminator is another LSTM aimed at distinguishing between the original video and its reconstruction from the summarizer. The summarizer LSTM is cast as an adversary of the discriminator, i.e., trained so as to maximally confuse the discriminator. This learning is also regularized for sparsity. Evaluation on four benchmark datasets, consisting of videos showing diverse events in first- and third-person views, demonstrates our competitive performance in comparison to fully supervised state-of-the-art approaches.

4.1 Introduction

A wide range of applications require automated summarization of videos [3, 185], e.g., for saving time of human inspection, or enabling subsequent video analysis. Depending on the application, there are various distinct definitions of video summarization [127, 120, 121, 7, 164, 139, 44, 89, 79, 65, 118, 56]. In this paper, we consider unsupervised video summarization, and cast it as a key frame selection problem. Given a sequence of video frames, our goal is to select a sparse subset of frames such that a representation error between the video and its summary is minimal.

Our problem statement differs from other formulations considered in the literature, for example, when a particular domain of videos to be summarized is a priori known (e.g., first-person



(a) (b)

Figure 4.1: (a) Overview: Our goal is to select key frames such that a distance between feature representations of the selected key frames and the video is minimized. (b) As specifying a suitable distance between deep features is difficult, we use a generative adversarial framework for optimizing the frame selector. Our approach consists of a variational auto-encoder and a generative adversarial network.

videos) [79], or when ground-truth annotations of key frames are provided in training data based on attention, aesthetics, quality, landmark presence, and certain object occurrences and motions [52].

Fig. 4.1a shows an overview of our approach to selecting key frames from a given video. The key frame selector is learned so as to minimize a distance between features extracted from the video and the selected key frames. Following recent advances in deep learning [138, 178, 186], we extract deep features from both the video and selected sequence of key frames using a cascade of a Convolutional Neural Network (CNN) – specifically GoogleNet [142] – and Long Short-Term Memory Network (LSTM) [58, 138]. The CNN is grounded onto pixels and extracts deep features from a given frame. The LSTM then fuses a sequence of the CNN's outputs for capturing long-range dependencies among the frames, and produces its own deep feature representing the input sequence. Specifically, we use the (variational) auto-encoder LSTM [138, 76] as a suitable deep architecture for unsupervised learning of video features. Given a distance between the deep representations of the video and selected key frames, our goal is to optimize the frame selector such that this distance is minimized over training examples.

Recent work, however, demonstrates that specifying a suitable distance of deep features is difficult [87]. Hence, we resort to the generative adversarial framework [48], which extends the aforementioned video summarization network with an additional discriminator network. As shown in Fig. 4.1b, the decoder part of the summarizer is used to reconstruct a video from the sequence of selected key frames. Then, we use a discriminator, which is another LSTM, to distinguish between the original video and its reconstruction from the summarizer. The auto-encoder LSTM and the frame selector are jointly trained so as to maximally confuse the discriminator LSTM – i.e., they are cast in a role of the discriminator's adversary – such that the discriminator has a high error rate in recognizing between the original and reconstructed videos. When this recognition error becomes maximum, we deem that the frame selector is learned to produce optimal video summarizations.

As we will show in this paper, our approach allows for an effective regularization of generative-adversarial learning in terms of: (i) limiting the total number of key frames that can be selected; or (ii) maximizing visual diversity among the selected key frames. For a fair comparison with related approaches to fully supervised video summarization – a different setting from ours that provides access to ground-truth key frame annotations in training – we also show how to effectively incorporate the available supervision as an additional type of regularization in learning.

Evaluation on four benchmark datasets, consisting of videos showing diverse events in first- and third-person views, demonstrates our competitive performance in comparison to fully supervised state-of-the-art approaches.

Our contributions include:

1. A new approach to unsupervised video summarization that combines variational auto-encoders and generative-adversarial training of deep architectures.
2. First specification of generative-adversarial training on high resolution video sequences.

In the following, Sec. 4.2 reviews prior work, Sec. 4.3 briefly introduces the generative adversarial network (GAN) and the variational autoencoder (VAE) models, Sec. 4.4 specifies main components of our approach, Sec. 4.5 formulates our end-to-end training, Sec. 4.6 describes variants of our approach differing in types of regularization we use in learning, and finally Sec. 4.7 presents our results.

4.2 Related Work

This section reviews related: (i) problem formulations of video summarization; (ii) approaches to supervised and unsupervised video summarization; (iii) deep learning approaches; and (iv) work using the generative adversarial framework in learning.

Various Problem Formulations. Video summarization is a long-standing problem, with various formulations considered in the literature. For example, the video synopsis [121] tracks moving objects, and then packs the identified video tubes into a smaller space-time volume. Also, montages [7, 164, 139] merge and overlaps key frames into a single summary image. Both of these problem formulations, however, do not require that the video summary preserves the information about a temporal layout of motions in the video. Previous work has also studied hyperlapses where the camera viewpoint is being changed during the time-lapse for speeding-up or slowing-down certain parts of the input video [79, 65, 118, 56]. Our problem statement is closest to storyboards, representing a subset of representative video frames [44, 89]. However, except for [186, 178], existing approaches to generating storyboards do not take advantage of deep learning.

Supervised vs. Unsupervised Summarization. Supervised methods assume access to human annotations of key frames in training videos, and seek to optimize their frame selectors so as to minimize loss with respect to this ground truth [45, 186, 185]. However, for a wide range of domains, it may be impossible to provide reliable and a sufficiently large amount of human annotations (e.g., military, nursing homes). These domains have been addressed with unsupervised methods, which typically use heuristic criteria for ranking and selecting key frames [92, 178, 74, 189, 136]. There have been attempts to use transfer learning for domains without supervision [186], but the surprisingly better performance of the transfer learning setting compared to the canonical setting, reported in [186], suggests a high correlation of the domains for three training dataset and one test dataset , which is hard to ensure in real-world settings.

Deep Architectures for Video Summarization. In [186], two LSTMs are used – one along the time sequence and the other in reverse from the video's end – to select key video frames, and trained by minimizing the cross-entropy loss on annotated ground-truth key frames with an additional objective based on determinantal point process (DPP) to ensure diversity of the selected frames. Our main differences are that we do not consider the key frame annotations, and train our LSTMs using the unsupervised generative-adversarial learning. In [178], recurrent auto-encoders are learned to represent annotated temporal intervals in training videos, called highlights. In contrast, we do not require human annotations of highlights in training, and we do not perform temporal video segmentation (highlight vs non-highlight), but key frame selection.

Generative Adversarial Networks (GANs) have been used for image-understanding problems [48, 124, 131, 128], and frame prediction/generation [106, 149, 42]. But we are not aware of their previous use for video summarization. In [87], the discriminator output of a GAN is used to provide a learning signal for the variational auto-encoder (VAE). We extend this approach in three critical ways: (1) We specify a new variational auto-encoder LSTM, whereas their auto-encoder is not a recurrent neural network, and thus cannot be used for videos; (2) Our generative-adversarial learning additionally takes into account the frame selector – a component not considered in [87]; and (3) We formulate regularization of generative-adversarial learning that is suitable for video summarization.

4.3 Review of VAE and GAN

Variational Autoencoder (VAE) [76] is a directed graphical model which defines a posterior distribution over the observed data, given an unobserved latent variable. Let $\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})$ be a prior over the unobserved latent variable, and $\boldsymbol{x}$ be the observed data. One can interpret $\boldsymbol{z}$ as the encoding of $\boldsymbol{x}$ and define $q(\boldsymbol{z}|x)$ as the probability of observing $\boldsymbol{z}$ given $\boldsymbol{x}$. It is typical to set $p_{\boldsymbol{z}}(\boldsymbol{z})$ as the standard normal distribution. Similarly, $p(\boldsymbol{x}|\boldsymbol{z})$ identifies the conditional generative distribution for $\boldsymbol{x}$. Learning is done by minimizing the negative log-likelihood of the data distribution:

$$-\log \frac{p(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})}{q(\boldsymbol{z}|\boldsymbol{x})} = \underbrace{-\log(p(\boldsymbol{x}|\boldsymbol{z}))}_{\mathcal{L}_{\text{reconst}}} + \underbrace{D_{KL}(q(\boldsymbol{z}|\boldsymbol{x})\|p(\boldsymbol{z})}_{\mathcal{L}_{\text{prior}}}. \qquad (4.1)$$

For efficient learning, Kingma et al. [76] propose a reparameterization of the variational lower bound suitable for stochastic gradient descent.

Generative Adversarial Network (GAN) [48] is a neural network that consists of two competing subnetworks: i) a 'generator' network (G) which generates data mimicking an unknown distribution and ii) a 'discriminator' network (D) which discriminates between the generated samples and the ones from true observations. The goal is to find a generator which fits the true data distribution while maximizing the probability of the discriminator making a mistake.

Let $\boldsymbol{x}$ be the true data sample, $\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})$ be the prior input noise, and $\hat{\boldsymbol{x}} = G(\boldsymbol{z})$ be the generated sample. Learning is formulated as the following minimax optimization:

$$\min_{G} \max_{D} \big[\underbrace{\mathbb{E}_{\boldsymbol{x}}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}}[\log(1 - D(\hat{\boldsymbol{x}}))]}_{\mathcal{L}_{\text{GAN}}} \big], \tag{4.2}$$

where D is trained to maximize the probability of correct sample classification (true vs generated) and G is simultaneously trained to minimize $\log(1 - D(\hat{\boldsymbol{x}}))$.

4.4 Main Components of Our Approach

Our approach consists of the summarizer and discriminator recurrent networks, as illustrated in Figure 4.2.

Given CNN's deep features for every frame of the input video, $\boldsymbol{x} = \{\boldsymbol{x}_t : t = 1, \ldots, M\}$, the summarizer uses a selector LSTM (sLSTM) to select a subset of these frames, and then an encoder LSTM (eLSTM) to encode the sequence of selected frames to a deep feature, $\boldsymbol{e}$. Specifically, for every frame $\boldsymbol{x}_t$, sLSTM outputs normalized importance scores $\boldsymbol{s} = \{s_t : s_t \in [0,1], t = 1, \ldots, M\}$ for selecting the frame. The input sequence of frame features $\boldsymbol{x}$ is weighted with these importance scores, and then forwarded to eLSTM. Note that in the special case of discretized scores, $s_t \in \{0,1\}$, eLSTM receives only a subset of frames for which $s_t = 1$. The last component of the summarizer is a decoder LSTM (dLSTM), which takes $\boldsymbol{e}$ as input, and reconstructs a sequence of features corresponding to the input video, $\hat{\boldsymbol{x}} = \{\hat{\boldsymbol{x}}_1, \hat{\boldsymbol{x}}_2, ..., \hat{\boldsymbol{x}}_M\}$.

The discriminator is aimed at distinguishing between $\boldsymbol{x}$ and $\hat{\boldsymbol{x}}$ as belonging to two distinct classes: 'original' and 'summary'. This classifier can be viewed as estimating a distance between $\boldsymbol{x}$ and $\hat{\boldsymbol{x}}$, and assigning distinct class labels to $\boldsymbol{x}$ and $\hat{\boldsymbol{x}}$ if their distance is sufficiently large. In this sense, the discriminator serves to estimate a representation error between the original video and our video summarization. While one way to implement the discriminator could be an energy-based encoder-decoder [190], in our experiments a binary sequence classifier have shown better performance. Hence, we specify the discriminator as a classifier LSTM (cLSTM) with a binary-classification output.

Analogous to the generative adversarial networks presented in [48, 87], we have that dLSTM and cLSTM form the generative adversarial network (GAN). The summarizer and discriminator

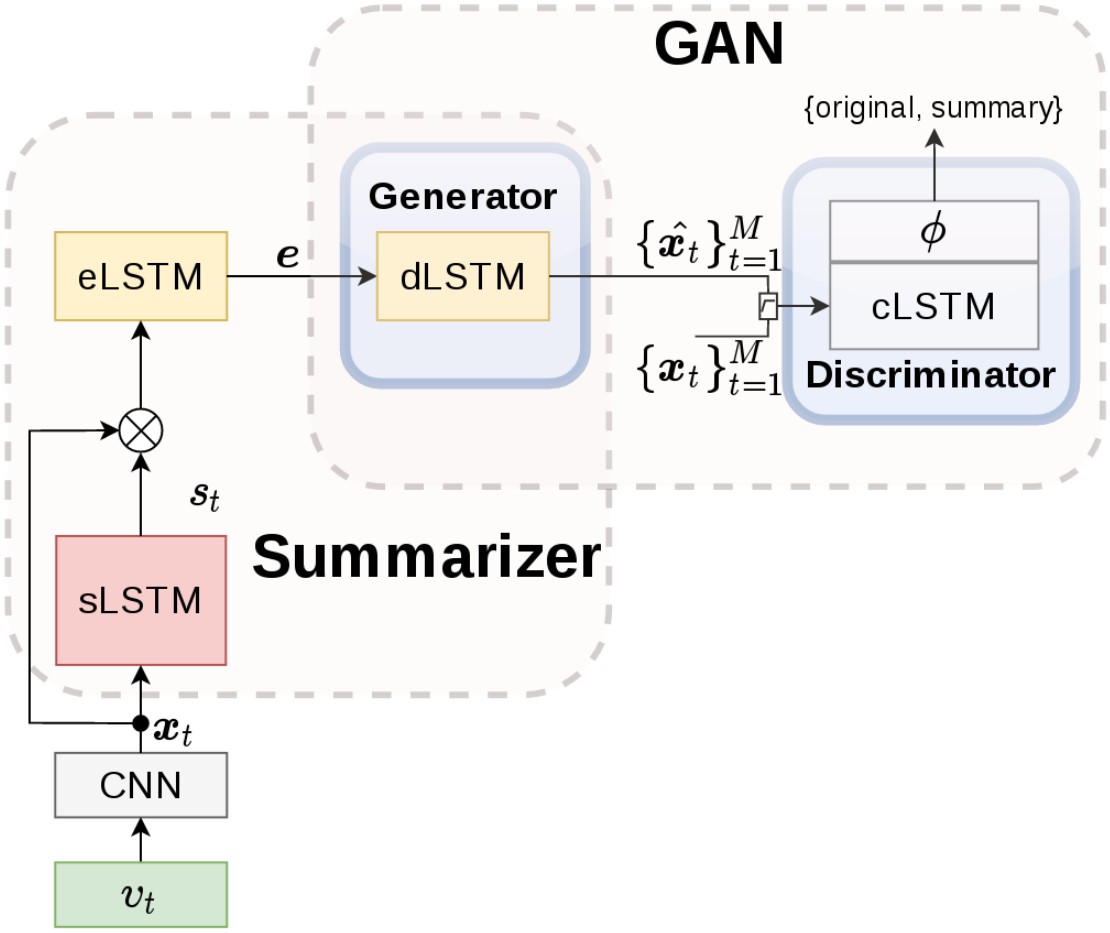


Figure 4.2: Main components of our approach: The selector LSTM (sLSTM) selects a subset of frames from the input sequence $\boldsymbol{x}$. The encoder LSTM (eLSTM) encodes the selected frames to a fixed-length feature $\boldsymbol{e}$, which is then forwarded to the decoder LSTM (dLSTM) for reconstructing a video $\hat{\boldsymbol{x}}$. The discriminator LSTM (cLSTM) classifies $\hat{\boldsymbol{x}}$ as 'original' or 'summary' class. dLSTM and cLSTM form the generative adversarial network (GAN).

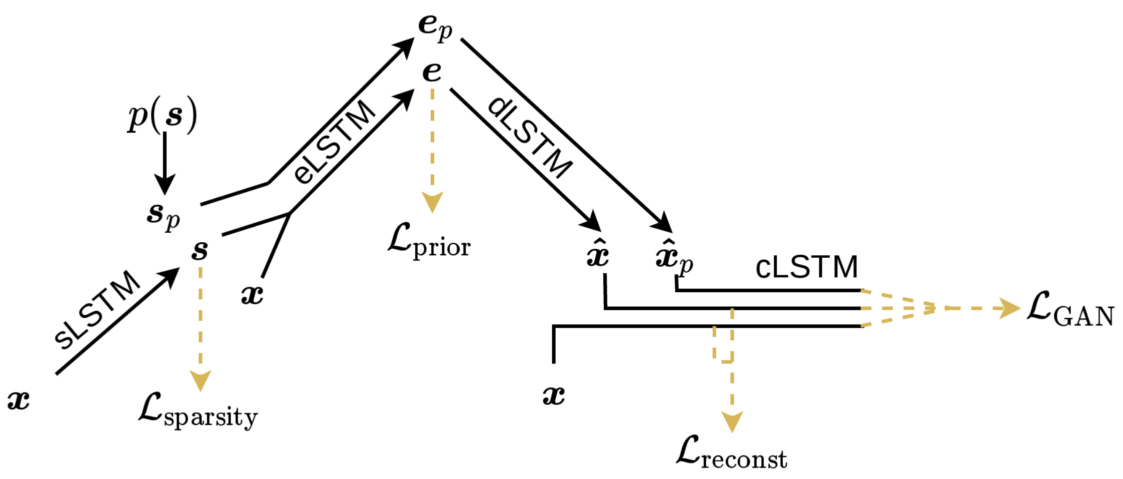

e_p
e
$p(s)$
s_p
s
x
eLSTM
dLSTM
sLSTM
$\mathcal{L}_{\text{prior}}$
$\hat{x}$
$\hat{x}_p$
cLSTM
$\mathcal{L}_{\text{GAN}}$
$\mathcal{L}_{\text{sparsity}}$
$\mathcal{L}_{\text{reconst}}$

Figure 4.3: The four loss functions used in our training. In training, we use an additional frame selector s_p, governed by a prior distribution (e.g., uniform), which produces the encoded representation e_p, and reconstruction $\hat{x}_p$. The adversarial training of cLSTM is regularized such that it is highly accurate on recognizing $\hat{x}_p$ as 'summary', but that it confuses $\hat{x}$ as 'original'.

networks are trained adversarially until the discriminator is not able to discriminate between the reconstructed videos from summaries and the original videos.

4.5 Training of sLSTM, eLSTM, and dLSTM

This section specifies our learning of: (i) Summarizer parameters, $\{\theta_s, \theta_e, \theta_d\}$, characterizing sLSTM, eLSTM, and dLSTM; and (ii) GAN parameters, $\{\theta_d, \theta_c\}$, defining dLSTM and cLSTM. Note that θ_d are shared parameters between the summarizer and GAN.

As illustrated in Fig. 4.3, our training is defined by four loss functions: 1) Loss of GAN, $\mathcal{L}_{\text{GAN}}$, 2) Reconstruction loss for the recurrent encoder-decoder, $\mathcal{L}_{\text{reconst}}$, 3) Prior loss, $\mathcal{L}_{\text{prior}}$, and 4) Regularization loss, $\mathcal{L}_{\text{sparsity}}$. The key idea behind our generative-adversarial training is to introduce an additional frame selector s_p, governed by a prior distribution (e.g., uniform distribution), $s_p \sim p(s_p)$. Sampling the input video frames with s_p gives a subset which is passed to eLSTM, producing the encoded representation e_p. Given e_p, dLSTM reconstructs a video sequence $\hat{x}_p$. We use $\hat{x}_p$ to regularize learning of the discriminator, such that cLSTM is highly accurate on recognizing $\hat{x}_p$ as the 'summary' class, but that it confuses $\hat{x}$ as 'original' class. Recall that the $\mathcal{L}_{\text{prior}}$ is imposed by the prior distribution over e as in (4.1).

Similar to the training of GAN models in [48, 87], we formulate an adversarial learning algorithm that iteratively optimizes the following three objectives:

1. For learning $\{\theta_s, \theta_e\}$, minimize $(\mathcal{L}_{\text{reconst}}+\mathcal{L}_{\text{prior}}+\mathcal{L}_{\text{sparsity}})$.
2. For learning θ_d, minimize $(\mathcal{L}_{\text{reconst}}+\mathcal{L}_{\text{GAN}})$.
3. For learning θ_c, maximize $\mathcal{L}_{\text{GAN}}$.

In the following, we define $\mathcal{L}_{\text{reconst}}$ and $\mathcal{L}_{\text{GAN}}$, while the specification of $\mathcal{L}_{\text{sparsity}}$ is deferred to Sec. 4.6.

Reconstruction loss $\mathcal{L}_{\text{reconst}}$: The standard practice in learning encoder-decoder networks is to use the Euclidean distance between the input and decoded output, $\|\boldsymbol{x}-\hat{\boldsymbol{x}}\|_2$, for estimating the reconstruction error. However, recent findings demonstrate shortcomings of this practice [87]. Hence, instead, we define $\mathcal{L}_{\text{reconst}}$ based on the hidden representation in cLSTM – specifically, the output of the last hidden layer of cLSTM, $\phi(\boldsymbol{x})$, for input $\boldsymbol{x}$. Note that while $\boldsymbol{x}$ is a sequence of features, $\phi(\boldsymbol{x})$ represents a compact feature vector, capturing long-range dependencies in the input sequence. Therefore, it seems more appropriate to use $\phi(\boldsymbol{x})$, rather than $\boldsymbol{x}$, for specifying $\mathcal{L}_{\text{reconst}}$.

Specifically, we formulate $\mathcal{L}_{\text{reconst}}$ as an expectation of a log-likelihood $\log p(\phi(\boldsymbol{x})|\boldsymbol{e})$, given that $\boldsymbol{x}$ has been passed through the frame selector $\boldsymbol{s}$ and eLSTM, resulting in $\boldsymbol{e}$:

$$\mathcal{L}_{\text{reconst}} = \mathbb{E}[-\log p(\phi(x)|\boldsymbol{e})], \tag{4.3}$$

where expectation $\mathbb{E}$ is approximated as the empirical mean of training examples. In this paper, we consider $p(\phi(\boldsymbol{x})|\boldsymbol{e})) \propto \exp(-\|\phi(\boldsymbol{x}) - \phi(\hat{\boldsymbol{x}})\|^2)$, while other non-Gaussian likelihoods are also possible.

Loss of GAN, $\mathcal{L}_{\text{GAN}}$: Following [87], our goal is to train the discriminator such that cLSTM classifies reconstructed video summaries $\hat{\boldsymbol{x}}$ as 'summary' and original sequences $\boldsymbol{x}$ as 'original'. In order to regularize this training, we additionally enforce that cLSTM learns to classify randomly generated summaries $\hat{\boldsymbol{x}}_p$ as 'summary', where $\hat{\boldsymbol{x}}_p$ is reconstructed from a subset of video frames randomly selected by sampling from a given prior distribution. In this paper, for this prior, we consider the uniform distribution. This gives:

$$\begin{aligned}\mathcal{L}_{\text{GAN}} = \quad & \log(\text{cLSTM}(\boldsymbol{x})) + \log(1 - \text{cLSTM}(\hat{\boldsymbol{x}})) \\ & + \log(1 - \text{cLSTM}(\hat{\boldsymbol{x}}_p)),\end{aligned} \tag{4.4}$$

where cLSTM$(\cdot)$ denotes the binary soft-max output of cLSTM.

Given the above definitions of $\mathcal{L}_{\text{reconst}}$ and $\mathcal{L}_{\text{GAN}}$, as well as $\mathcal{L}_{\text{sparsity}}$ explained in Sec. 4.6, we update the parameters θ_s, θ_e, θ_d and θ_c using the Stochastic Gradient Variational Bayes estimation [77, 76], adapted for recurrent networks [29]. Algorithm 4 summarizes all steps of our training. Note that Algorithm 4 uses capital letters to denote a mini-batch of the corresponding variables with small-letter notation in the previous text.

Algorithm 2 Training SUM/GAN model

```
1:  Input: Training video sequences
2:  Output: Learned parameters $\{\theta_s, \theta_e, \theta_d, \theta_c\}$.
3:  Initialize all parameters $\{\theta_s, \theta_e, \theta_d, \theta_c\}$
4:  for max number of iterations do
5:      $X \leftarrow$ mini-batch from CNN feature sequences
6:      $S \leftarrow$ sLSTM$(X)$ % select frames
7:      $E =$ eLSTM$(X, S)$ % encoding
8:      $\hat{X} =$ dLSTM$(E)$ % reconstruction
9:      $S_p \leftarrow$ draw samples form the uniform distribution
10:     $E_p =$ eLSTM$(X, S_p)$ % encoding
11:     $X_p =$ dLSTM$(E_{S_p})$ % reconstruction
12:     % Updates using Stochastic Gradient:
13:     $\{\theta_s, \theta_e\} \overset{+}{\leftarrow} -\nabla(\mathcal{L}_{\text{reconst}} + \mathcal{L}_{\text{prior}} + \mathcal{L}_{\text{sparsity}})$
14:     $\{\theta_d\} \overset{+}{\leftarrow} -\nabla(\mathcal{L}_{\text{reconst}} + \mathcal{L}_{\text{GAN}})$
15:     $\{\theta_c\} \overset{+}{\leftarrow} +\nabla(\mathcal{L}_{\text{GAN}})$ % maximization update
16: end for
```

4.6 Variants of our Approach

This section explains our regularization of learning. We use the following three types of regularization, which define the corresponding variants of our approach.

Summary-Length Regularization penalizes having a large number of key frames selected in the summary as:

$$\mathcal{L}_{\text{sparsity}} = \left\| \frac{1}{M} \sum_{t=1}^{M} s_t - \sigma \right\|_2 \tag{4.5}$$

where M is the total number of video frames, and σ is an input hyper-parameter representing a percentage of frames that we expect to be selected in the summary. When our approach uses $\mathcal{L}_{\text{sparsity}}$, we call it SUM-GAN.

Diversity Regularization enforces selection of frames with high visual diversity, in order to mitigate redundancy in the summary. In this paper, we use two standard definitions for diversity regularization – namely, (i) Determinantal Point Process (DPP) [142, 45, 186]; and (ii) Repelling regularizer (REP) [190].

Following [186], our DPP based regularization is defined as:

$$\mathcal{L}^{\text{dpp}}_{\text{sparsity}} = -\log(P(\boldsymbol{s})) \tag{4.6}$$

where $P(\boldsymbol{s})$ is a probability that DPP assigns to the selection indicator $\boldsymbol{s}$. We compute $P(\boldsymbol{s}; L) = \frac{\det(L(\boldsymbol{s}))}{\det(L+I)}$, where L is an $M \times M$ similarity matrix between every two hidden states in eLSTM, I is an identity matrix and $L(\boldsymbol{s})$ is a smaller square matrix, cut down from L given $\boldsymbol{s}$. Let $\boldsymbol{e}_t$ be the hidden state of eLSTM at time t. For time steps t and t' the pairwise similarity values are defined as $L_{t,t'} = s_t s_{t'} \boldsymbol{e}_t \boldsymbol{e}_{t'}$.

When our approach uses $\mathcal{L}^{\text{dpp}}_{\text{sparsity}}$, we call it SUM-GAN$_{\text{dpp}}$.

For repelling regularization, we define

$$\mathcal{L}^{\text{rep}}_{\text{sparsity}} = \frac{1}{M(M-1)} \sum_{t} \sum_{t' \neq t} \left(\frac{\boldsymbol{e}_t^{\top} \boldsymbol{e}_{t'}}{\|\boldsymbol{e}_t\| \|\boldsymbol{e}_{t'}\|} \right) \tag{4.7}$$

and call this variant of our approach as SUM-GAN$_{\text{rep}}$.

Keyframe Regularization is specified for the supervised setting where ground-truth annotations of key frames are provided in training. This regularization enables a fair comparison of our approach with recently proposed supervised methods. Note that we here consider importance scores as 2D softmax outputs $\{\boldsymbol{s}_t\}$, rather than scalar values as introduced in Sec. 4.4. We define the sparsity loss as the cross-entropy loss:

$$\mathcal{L}_{\text{sparsity}}^{\text{sup}} = \frac{1}{M} \sum_{t} \text{cross-entropy}(\boldsymbol{s}_t, \hat{\boldsymbol{s}}_t). \tag{4.8}$$

We call this variant of our approach as SUM-GAN$_{\text{sup}}$.

4.7 Results

Datasets. We evaluate our approach on four datasets: SumMe [53], TVSum [136], Open Video Project (OVP) [2, 22], and Youtube [22]. 1) SumMe consists of 25 user videos. The videos capture multiple events such as cooking and sports. The video contents are diverse and include both first-person and third-person camera. The video lengths vary from 1.5 to 6.5 minutes. The dataset provides frame-level importance scores. 2) TVSum contains 50 videos from YouTube. The videos are selected from 10 categories in the TRECVid Multimedia Event Detection (MED) (5 videos per category). The video lengths vary from 1 to 5 minutes. Similar to SumMe, the video contents are diverse and include both ego-centric and third-person camera. 3) For OVP, we evaluate on the same 50 videos used in [22]. The videos are from various genres (e.g. documentary, educational) and their lengths vary from 1 to 4 minutes. 4) The YouTube dataset includes 50 videos collected from websites. The duration of the videos are from 1 to 10 minutes and the content include cartoons, news and sports.

Evaluation Setup. For a fair comparison with the state of the art, the keyshot-based metric proposed in [186] is used for evaluation. Let A be the generated keyshots and B the user-annotated keyshots. The precision and recall are defined based on the amount of temporal overlap between A and B as follows:

$$\begin{aligned} \text{precison} &= \frac{\text{duration of overlap between } A \text{ and } B}{\text{duration of } A} \\ \text{recall} &= \frac{\text{duration of overlap between } A \text{ and } B}{\text{duration of } B} \end{aligned} \tag{4.9}$$

Finally, the harmonic mean F-score is used as the evaluation metric:

$$\mathrm{F_{score}} = 2 \cdot \frac{\text{precison} \times \text{recall}}{\text{precison} + \text{recall}} \tag{4.10}$$

We follow the steps in [186] to convert frame-level scores to key frames and key shot summaries, and vice versa in all datasets. To generate key shots for datasets which only provide key frame scores, the videos are initially temporally segmented into disjoint intervals using KTS [119]. The resulting intervals are ranked based on their importance score where the importance score of an interval is equal to the average score of the frames in that interval. A subset of intervals are selected from the ranked intervals as keyshots such that the total duration of the generated key shots are less than 15% of the duration of the original video.

For datasets with multiple human annotations (in the form of key shots or key frames), we follow the standard approach described in [54, 136, 186] to create a single ground-truth set for evaluation. While evaluating our SUM-GAN$_{\text{sup}}$ model, we used the same train, test and validation split as in [186]. For fair comparison, we run it for five different random splits and report the average performance.

Implementation Details: For fair comparison with [186], we choose to use the output of pool5 layer of the GoogLeNet network [142] (1024-dimensions), trained on ImageNet [130], for the feature descriptor of each video frame. We use a two-layer LSTM with 1024 hidden units at each layer for discriminator LSTM (cLSTM). We use two two-layer LSTMs with 2048 hidden units at each layer for eLSTM and dLSTM respectively. It is shown in [138] that a decoder LSTM which attempts to reconstruct the reverse sequence is easier to train. Similarly, our dLSTM reconstruct the feature sequence in the reverse order. Note that while presenting $\boldsymbol{x}$ and $\hat{\boldsymbol{x}}$ as the cLSTM input, both sequences should have similar ordering in time.

We initialize the parameters of eLSTM and dLSTM, with the parameters of a pre-trained recurrent autoencoder model trained on feature sequences from original videos. We find out that this helps to improve the overall accuracy and also results in faster convergence.

The sLSTM network is a two-layer bidirectional LSTM with 1024 hidden units. The output is a 2-dimensional softmax layer in the case of SUM-GAN$_{\text{sup}}$.

Baselines: It is important to point out that considering the generative structure of our approach and the definition of the update rules in Alg. 4, it is not possible to replace subnetworks of our model with baselines. In addition to different variations of our approach defined in sec. 4.6, we also define SUM-GAN$_{\text{bas}}$ which does not use the sparsity regularization.

Method	SumMe	TVSum	OpenVideo	YouTube
SUM-GAN	38.7	50.8	71.5	58.9
SUM-GAN$_{\text{bas}}$	35.7	50.1	69.8	57.1
SUM-GAN$_{\text{rep}}$	38.5	51.9	72.3	59.6
SUM-GAN$_{\text{dpp}}$	39.1	51.7	72.8	60.1
SUM-GAN$_{\text{sup}}$	41.7	56.3	77.3	62.5

Table 4.1: Comparison of different variations of our generative video summarization on benchmark datasets. The result for SUM-GAN is reported for $\sigma = 0.3$.

4.7.1 Quantitative Results

Table 4.1 summarizes the accuracy of different variations of our approach. As is expected, the model with additional frame-level supervision, SUM-GAN$_{\text{sup}}$, outperforms the unsupervised variants by (2-5%).

One interesting observation is that although explicit regularization of the model with 'diversity regularizers' (SUM-GAN$_{\text{dpp}}$ and SUM-GAN$_{\text{rep}}$) performs slightly better than the variant of our model with 'length regularizer' (SUM-GAN) the difference is not statistically significant. Furthermore, in the case of SumMe, SUM-GAN performs better than SUM-GAN$_{\text{rep}}$. This is particularly important because it verifies our main hypothesis that a good summary should include a subset of frames which provide similar content representation as of the original frame sequences. This suggests that if we constrain the summary to be shorter in length, implicitly the frames will be diverse. We also observe that SUM-GAN$_{\text{dpp}}$ performs better than SUM-GAN$_{\text{rep}}$ in all four datasets. We believe that this is mainly because of the fact that unlike the repelling regularizer, DPP is non-linear and can reinforce stronger regularization.

We are particularly interested in comparing our performance in contrast with prior unsupervised and supervised methods. This comparison is presented in table 4.2. As shown, our unsupervised SUM-GAN$_{\text{dpp}}$ model outperforms all unsupervised approaches in all datasets. For SumMe, our approach is almost 5% better than the state-of-the-art unsupervised approaches. More importantly, the accuracy of SUM-GAN$_{\text{dpp}}$ is comparably close to the supervised methods in TVSum, OVP and YouTube datasets.

Comparing with the state-of-the-art supervised approaches, our supervised variant, SUM-GAN$_{\text{sup}}$, outperforms in all datasets except OVP. Even in the case of OVP, we are statistically close to the best reported accuracy with 0.4% margin. We hypothesize that the accuracy boost

Learning	Method	SumMe	TVSum	OpenVideo	YouTube
unsupervised	[22]	33.7	-	70.3	59.9
	[92]	26.6	-	-	-
	[74]	-	36.0	-	-
	[136]	26.6	50.0	-	-
	[39]	-	-	63.4	-
	[109]	-	-	57.6	-
	[189]	-	46.0	-	-
	SUM-GAN$_{\text{dpp}}$	**39.1**	**51.7**	**72.8**	**60.1**
(semi-)supervised	[54]	39.7	-	-	-
	[185]	40.9	-	76.6	60.2
	[53]	39.3	-	-	-
	[186]	38.6	54.7	-	-
	[45]	-	-	**77.7**	60.8
	SUM-GAN$_{\text{sup}}$	**41.7**	**56.3**	77.3	**62.5**

Table 4.2: Comparison of our proposed video summarization approach compared to state of the art. The reported results from the state of the art are from published results. Note that [185, 45] use only 39 videos of non-cartoon videos.

is mainly because of the additional learning signal from the cLSTM. Note that the discriminator observes a longer sequence and classifies based on a learned semantic representation of the feature sequence. This enables the discriminator to provide a more informative signal regarding the importance of the frames for content similarity.

Zhang et al. [186] augment the SumMe and TVSum datasets with OVP and YouTube datasets and improve the accuracy on SumMe and TVSum. Table 4.3 shows the accuracy results in comparison with results reported in [186] when training dataset is augmented. Except for SUM-GAN$_{\text{sup}}$, which we use 80% of the target dataset in training, for the unsupervised variants of our approach we use all four datasets in training. The most important observation is that one of our unsupervised variations, SUM-GAN$_{\text{dpp}}$, outperforms the state of the art in SumMe. This shows that if trained with more unsupervised video data, our model is able to learn summaries which are competitive with the models trained using key frame annotations.

Finally, we evaluate the performance of our approach for different percentages of σ values for our SUM-GAN model. Fig. 4.4 shows the resulting F-score values for different σ's on four

Method	SumMe	TVSum
[185]	40.9	-
[186]	42.9	59.6
SUM-GAN	41.7	58.9
SUM-GAN$_{\text{rep}}$	42.5	59.3
SUM-GAN$_{\text{dpp}}$	43.4	59.5
SUM-GAN$_{\text{sup}}$	43.6	61.2

Table 4.3: Comparison of different variations of our generative video summarization with the state of the art for SumMe and TVSum datasets when the training data is augmented with additional data from OVP and YouTube datasets.

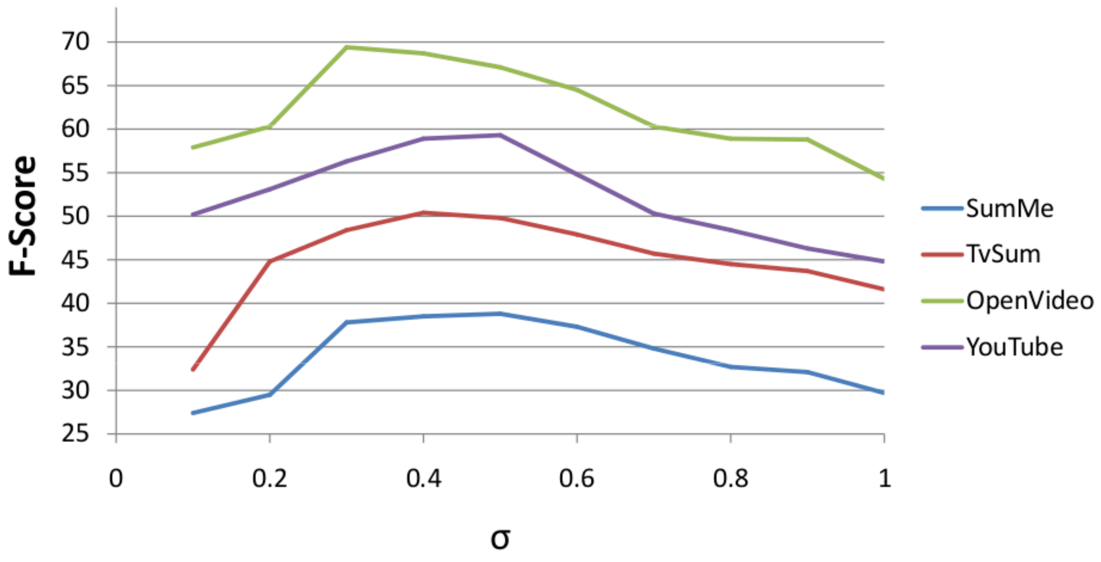
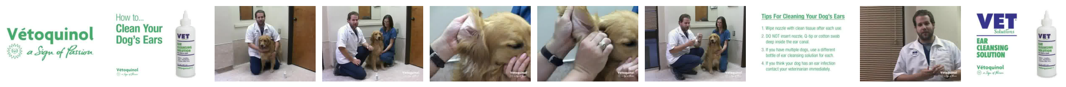
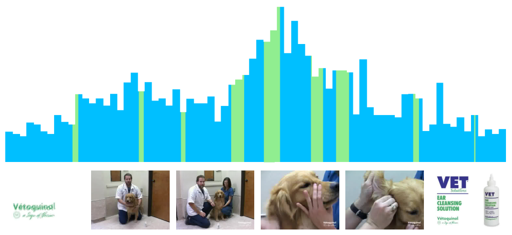


Figure 4.4: F-score results for different values of σ on SumMe, TvSum, OpenVideo, and YouTube.

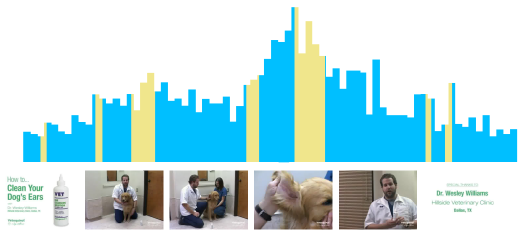
different datasets. While the performance is consistent for $0.3 \leq \sigma \leq 0.5$, it drops rapidly as $\sigma \to 1$ or $\sigma \to 0$.



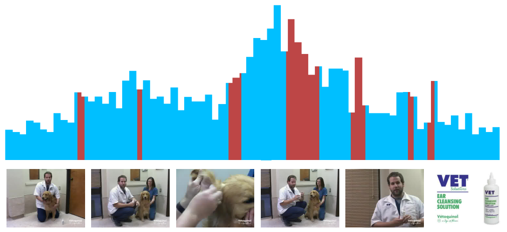
(a) Sample frames from video 15 (indexed as in [136])



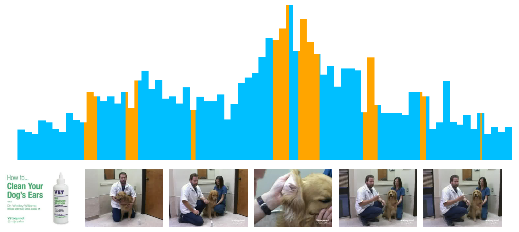
(b) SUM-GAN



(c) SUM-GAN$_{\text{rep}}$



(d) SUM-GAN$_{\text{dpp}}$



(e) SUM-GAN$_{\text{sup}}$

Figure 4.5: Example summaries from a sample video in TvSum [136]. The blue bars show the annotation importance scores. The colored segments are the selected subset of frames using the specified method.

4.7.2 Qualitative Results

To better illustrate the temporal selection pattern of different variations of our approach, we demonstrate the selected frames on an example video in Fig. 4.5. The blue background shows the frame-level importance scores. The colored regions are the selected subsets for different methods. The visualized key frames for different variants supports the result presented in Table 4.1. Despite small variations, all four approaches cover the temporal regions with high frame-level score.

4.7.3 Comparison with Shallow Features

We verified the generalizability of our video summarization approach to non-deep features by evaluating our model with the shallow features employed in [185, 186]. Table 4.4 shows the performance of our model compared to the state-of-the-art models which use shallow features.

Method	SumMe	TVSum
[136]	-	50.0
[185]	-	60.0
[186]	38.1	54.0
SUM-GAN	37.8	53.2
SUM-GAN$_{\text{rep}}$	38.8	54.1
SUM-GAN$_{\text{dpp}}$	41.2	53.9
SUM-GAN$_{\text{sup}}$	39.5	59.5

Table 4.4: Comparison of different variations of our generative video summarization with the state of the art for SumMe and TVSum datasets when using shallow features.

Besides the reported results in [185] for TvSum, where the shallow features outperform the deep features, our model consistently performs better the state of the art. Unlike [185], our model grounded on deep features still performs better than the same model grounded on shallow features.

4.8 Summary

We propose a generative architecture based on variational recurrent auto-encoders and generative adversarial networks for unsupervised video summarization to select a subset of key frames. The main hypothesis is that the learned representation of the summary video and the original video should be similar. The summarizer aims to summarize the video such that the discriminator is fooled and the discriminator aims to recognize the summary videos from original videos. The entire model is trained in an adversarial manner where the GAN's discriminator is used to learn a discrete similarity measure for training the recurrent encoder/decoder and the frame selector LSTMs. Variations of our approach are defined using different regularizations. Evaluation on benchmark datasets show that all unsupervised variations of our approach outperform the state of the art in video summarization by 2-5$\%$ and provides a comparable accuracy to the state-of-the-art supervised approaches. We also verified that the supervised variation of our approach outperforms the state of the art by 1-4$\%$.

Chapter 5: Approximate Policy Iteration for Budgeted Semantic Video Segmentation

Abstract

This paper formulates and presents a solution to the new problem of budgeted semantic video segmentation. Given a video, the goal is to accurately assign a semantic class label to every pixel in the video within a specified time budget. Typical approaches to such labeling problems, such as Conditional Random Fields (CRFs), focus on maximizing accuracy but do not provide a principled method for satisfying a time budget. For video data, the time required by CRF and related methods is often dominated by the time to compute low-level descriptors of supervoxels across the video. Our key contribution is the new budgeted inference framework for CRF models that intelligently selects the most useful subsets of descriptors to run on subsets of supervoxels within the time budget. The objective is to maintain an accuracy as close as possible to the CRF model with no time bound, while remaining within the time budget. Our second contribution is the algorithm for learning a policy for the sparse selection of supervoxels and their descriptors for budgeted CRF inference. This learning algorithm is derived by casting our problem in the framework of Markov Decision Processes, and then instantiating a state-of-the-art policy learning algorithm known as Classification-Based Approximate Policy Iteration. Our experiments on multiple video datasets show that our learning approach and framework is able to significantly reduce computation time, and maintain competitive accuracy under varying budgets.

5.1 Introduction

Motivation: The design of vision approaches is typically informed by a trade-off between efficiency and accuracy. Good computational efficiency is usually achieved by taking a number of heuristic pre- and post-processing steps and integrating them with the main approach. For example, vision practitioners heuristically limit the types of features to be extracted (e.g., low-cost ones), as well as locations and scales in images and video from which they are extracted. While these steps have been satisfactory for small-scale problems, their heuristic nature makes an adaptation of existing systems to settings with stringent runtime requirements very difficult.

Our goal is to formulate a principled framework for optimally adapting a vision system to varying time budgets imposed by particular application settings, so as to maximize overall performance for any budget and maintain an accuracy as close as possible to the system's performance with no time bound. In this paper, we focus our presentation of theory and experiments in the context of semantic video segmentation. Nevertheless, it is worth noting that our framework is based on fairly non-restrictive assumptions, and thus is suitable for many other computer vision problems, beyond the scope of this paper.

The Focus Vision Problem: Given a video, our goal is to assign a class label to every pixel from the set of semantic classes seen in training, under any time budget. We call this new problem *budgeted semantic video segmentation*. For example, in a video of a street, we want to efficiently segment spatiotemporal subvolumes occupied by cars, pedestrians, and buildings in less time than the user-specified bound. This is an important problem with a wide range of applications (e.g., driverless cars, sports video analytics) which require highly accurate and timely estimates of space-time extents of objects in the scene.

The key idea: We assume that a given approach to semantic video segmentation specifies an inference procedure (e.g., loopy belief propagation, graph-cut) which takes input features and outputs an inference result. Rather than pixels, most existing approaches, first, label supervoxels, obtained from an unsupervised (low-level) video partitioning, and then transfer these labels to the corresponding pixels. Thus, a typical input consists of supervoxels and their *descriptors*, where the descriptors may be computed locally at every supervoxel, between pairs of neighboring supervoxels, and globally across the entire video. Computing all descriptors for all supevoxels is costly. Thus, a sparse section of supervoxels and choosing their most useful and least costly descriptors is our key problem. Note that we do not modify the inference procedure. Rather, we adapt the descriptor computation step to suit budgeted settings of a broad family of approaches. Consequently, we do not seek to improve prediction accuracy, but rather maintain the existing level of accuracy while reducing runtime.

Our framework is illustrated in Fig. 5.1. We assume access to a Conditional Random Field (CRF) and associated inference procedure—one of the most popular formulations of semantic video segmentation—and design a *sequential* inference policy that interacts with the given approach. Given an unsupervised video partitioning into supervoxels and a user-specified time budget, our policy is presented with a sequence of candidate supervoxels until time expires and must decide for each one which new descriptor to run for it, if any, with the goal of maximizing

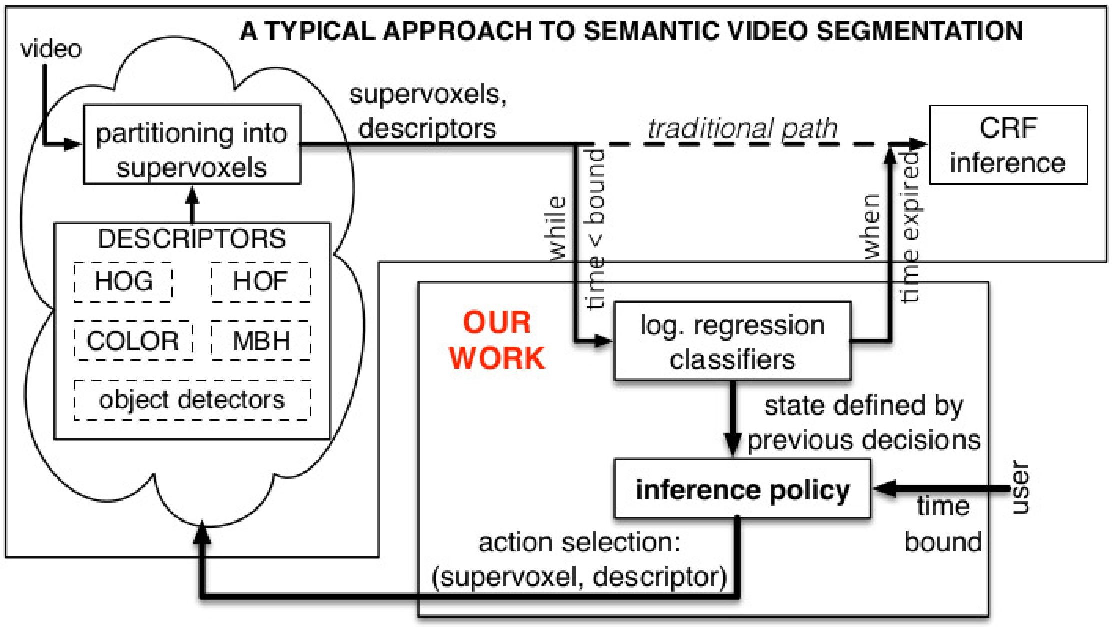

A TYPICAL APPROACH TO SEMANTIC VIDEO SEGMENTATION
video
partitioning into supervoxels
supervoxels, descriptors
traditional path
CRF inference
DESCRIPTORS
HOG
HOF
COLOR
MBH
object detectors
while time < bound
when time expired
OUR WORK
log. regression classifiers
state defined by previous decisions
inference policy
user
time bound
action selection: (supervoxel, descriptor)

Figure 5.1: We formalize a sequential inference policy aimed at adopting a fairly general family of approaches to the problem of budgeted semantic video segmentation. Our focus domain is a holistic CRF-based inference, but other approaches to semantic video segmentation could be considered. Given an unsupervised video partitioning into supervoxels, a set of feature extractors, and a user-specified time budget, our policy sequentially selects the best pair (supevoxel, feature) toward maximizing performance of the CRF inference until the time expires.

performance of the CRF inference. The sequential selection should take into account both the immediate and long-term value of the decisions toward overall inference accuracy.

We formalize such sequential decision making in the framework of Markov Decision Processes (MDPs), where, for a given state, the policy selects the highest value action among possible actions. The policy state is defined by the previous selections of descriptors for supervoxels, which is conveniently represented via special policy features. The policy actions include running a descriptor for the currently presented supervoxel or FINISHED, which specifies that no further descriptors should be computed for the current supervoxel. The policy is defined as a *linear* ranking function for balancing efficiency and expressiveness, such that its execution consumes resources negligibly, and that it captures sufficient information about the current policy state to support good decisions. For training the policy, we use the state-of-the-art policy learning approach of Classification-Based Approximate Policy Iteration (CAPI), which is able to leverage state-of-the-art classification learning techniques for policy optimization.

5.2 Related Work and Our Contributions

Semantic video segmentation is mostly formulated as a graphical-model based labeling of supervoxels in the video [15, 13, 12, 107, 16, 96, 143, 170]. For example, graphical models were used for: i) Propagating manual annotations of supervoxels of the first few frames to other supervoxels in the video [107, 15, 13], or ii) Supervoxel labeling based on week supervision in training [100]. The accuracy of such labeling can be improved by CRF-based reasoning about 3D relations [82] or context [96] among object classes in the scene. Therefore, it seems reasonable that we develop our framework for an existing CRF-based labeling of supervoxels. None of these methods explicitly studied their runtime efficiency, except for a few empirical results of sensitivity to the total number of supervoxels used [61].

Prior work has considered the issue of how to reduce inference runtime by specifying efficient approximations to the original inference [188, 80, 11]. Their work is not related to ours, since we keep the given inference procedure intact. A few approaches have addressed cost-sensitive inference for activity recognition [4, 6], image classification [69, 70], and object detection [172]. Our fundamental difference is that these methods precompute all features and then adapt the very inference procedure such that it uses only an optimal subset of the features to meet the time budget. In contrast, we do not compute any features before our inference policy makes the decision to do so.

Closely related work improves runtime efficiency by reducing the costs of feature extraction [162, 163, 129, 94, 50]. For example, CRF-based semantic scene labeling in images is made more efficient by computing only a small subset of unary potentials for the CRF, and efficiently predicting the missing potentials from neighbors [129]. This approach can be viewed as a special case of our framework, since they only select superpixels and then use *all* features for computing the unary potentials, whereas we would select both superpixels and feature types in their domain. In [163], a budget constrained reinforcement learning is used to select optimal features for tracking human poses in relatively simple videos. Since they extract a chosen feature over the entire video, this approach can be viewed as another special case of our framework, because we would additionally make the decision about where to extract the feature in the video. This is a crucial difference for large videos, where computing even a single feature/descriptor across an entire video may exceed the budget constraint. Additional differences from these two approaches are explained in Sec. 5.3. We efficiently compute low-level descriptors for a selected subset of supervoxels in addition to the object level potentials in [94]. Unlike [50] we do not change the inference module, which makes it possible to augment any state of the art with our approach. It is not clear how to extend [50] to use higher order potentials. More importantly our approach is able to exploit the high correlation among the features of neighboring supervoxels to avoid the extra feature extraction cost if it does not help the final inference.

Contributions: 1) First formalization of the budgeted semantic video segmentation problem. 2) Design of the first learning algorithm that can tune inference policies for varying time budgets. 3) Specification and evaluation of the inference policy representation as a linear function that supports learning.

5.3 Budgeted Semantic Video Segmentation

Our framework takes as input a time budget B and video that has been partitioned into a set of supervoxels V. The goal is to accurately assign a label to each supervoxel in V in time less than B. Below, we first present a common CRF formulation for semantic video segmentation, which our framework is based on. Next, we formulate our framework for bounding the CRF inference time.

5.3.1 A Common CRF Formulation

We consider a standard pairwise CRF model, which specifies the following score for any labeling $\{y_i\}$ of supervoxels $i \in V$ and their spatiotemporal neighborhood relationships $(i, j) \in E \subset V \times V$:

$$\sum_{i \in V} \boldsymbol{w}_u \cdot \boldsymbol{\psi}_u(\boldsymbol{x}_i, y_i) + \sum_{(i,j) \in E} \boldsymbol{w}_p \cdot \psi_p(\boldsymbol{x}_i, \boldsymbol{x}_j, y_i, y_j), \tag{5.1}$$

where $\boldsymbol{w}_u \cdot \boldsymbol{\psi}_u(\boldsymbol{x}_i, y_i)$ denotes the unary potential specified in terms of an n_u-dimensional *unary feature vector* $\boldsymbol{\psi}_u(\boldsymbol{x}_i, y_i)$ estimated for observation $\boldsymbol{x}_i$ at supervoxel i when i is assigned label y_i from the set of labels $\mathcal{L}$, and the corresponding weight vector $\boldsymbol{w}_u$. Also, we have that $\boldsymbol{w}_p \cdot \psi_p(\boldsymbol{x}_i, \boldsymbol{x}_j, y_i, y_j)$ assigns pairwise "compatibility" scores of assigning labels y_i to i and y_j to neighboring supervoxel j, where $\boldsymbol{\psi}_p(\boldsymbol{x}_i, y_i, \boldsymbol{x}_j, y_j)$ is an n_p-dimensional *pairwise feature vector* and $\boldsymbol{w}_p$ as the associated weight vector. We specify $n_u = L = |\mathcal{L}|$ and $n_p = L^2$, but more general specifications are also possible.

We consider a common form of unary features defined in terms of a probabilistic multi-class classifier. In particular, we will learn a probabilistic classifier H that returns an L-dimensional probability vector, such that $H(i)$ is a predicted distribution over all labels $y_i \in \mathcal{L}$ for supervoxel i. The input to H, used as a basis for prediction, is the concatenation of multiple descriptors of supervoxel i. In this paper, we use logistic regression for H. Given H, the unary feature vector $\boldsymbol{\psi}_u(\boldsymbol{x}_i, y_i)$ is constructed in the standard way by setting all elements of $\boldsymbol{\psi}_u(\boldsymbol{x}_i, y_i)$ to zero, except for the element corresponding to label y_i which is set to the probability $H(i)$ of predicting y_i for i. Thus, the cost of $\boldsymbol{\psi}_u(\boldsymbol{x}_i, y_i)$ is dominated by the cost of computing $H(i)$.

The pairwise features is specified in the standard way as the difference between descriptors of the pair of neighboring supervoxels. Thus, the pairwise "compatibility" score is defined to increase as this difference becomes smaller for the supervoxels with the same label, and to decrease for small descriptor differences when the pair of supervoxels have different labels. Non-overlapping features are replaced by the expected features conditioned on the label.

CRF inference involves two main steps. First, unary and pairwise potentials are computed for all combinations of supervoxels and labels. Second, a standard approximate CRF inference procedure is applied, such as belief propagation or α-expansion (used in our experiments), which returns a high scoring (ideally optimal) label assignment. The overall computational cost is, thus, the total time for computing the potentials and running the inference procedure.

Learning the CRF and H. Given labeled training data, we learn our CRF model, $\mathcal{M}$, using the unary and pairwise feature vectors computed over all training supervoxels. This is done by computing all descriptors for all supervoxels in the training data, and then using a standard CRF library to obtain $\mathcal{M}$ and the associated logistic regression H.

Recall, that the main idea of our framework is to reduce time by only computing a subset of descriptors for some supervoxels. Thus, in order to produce the unary feature vector, H must be able to make label predictions for a supervoxel using any subset of its descriptors. One approach to obtaining such predictions would be to train H on all descriptors, and then use one of a number of common strategies for making predictions with "missing information" (e.g., replacing the missing descriptor values with zero or expected values).

Rather, we take a more brute force approach, with the advantage of not requiring any method for handling missing descriptors. Instead of just training a single classifier based on all descriptors, we train a *collection of classifiers*, one classifier for each possible subset of descriptors. H is then represented by this collection of classifiers. That is, each logistic regression is trained by removing all descriptor information from the training set other than its assigned descriptor subset. When H is asked to make a prediction for a supervoxel during budgeted inference, it uses the classifier corresponding to the set of descriptors that have been run for that supervoxel. Given that classifier training is very fast, this will often be practical even with thousands of possible subsets. This is indeed the case in our experiments as described in Sec. 5.5.

5.3.2 Budgeted Unary Feature Computation

Computing the unary features ψ_u typically dominates overall computation time, since this involves computing a number of low-level descriptors over all supervoxels. The pairwise features ψ_p are much cheaper, in comparison, since they are generally based on comparing descriptors already computed for ψ_u. For this reason, our framework focuses on bounding the time of computing ψ_u, and we will let B denote this time bound for the remainder of this paper. The overall CRF inference will typically be a small constant larger than B, when B is non-trivial.

Our hypothesis is that similar accuracies in inference can be achieved with less cost by intelligently selecting for each supervoxel a sparse set of descriptors to compute, including the empty set, which are then used by H to generate ψ_u. Below, we describe how to make these decisions.

Alg. 3 presents our iterative approach to selecting descriptor subsets for time bounded inference. Throughout the iterations, we maintain a set of *candidate supervoxels*, $\mathcal{C}$, which are currently being considered for descriptor computation. $\mathcal{C}$ is initialized to a small random subset of V. We also maintain a set of *finished supervoxels*, $\mathcal{F}$, which will no longer be considered for descriptor computation. Each iteration consists of two steps. The first step calls the function Select($\mathcal{C}$), which returns a supervoxel $i \in \mathcal{C}$ to be considered next. As described in Sec. 5.5, we consider two versions of Select($\mathcal{C}$): a) Random selection, and b) Priority-based selection. The second step applies *policy* π for selecting either a new descriptor for i to compute, or mark i as being finished. Specifically, $\pi(i)$ returns an *action* for i that is either a descriptor index or FINISHED. In the latter case, i is moved from $\mathcal{C}$ to $\mathcal{F}$, and all neighbors of i that are not already in $\mathcal{F}$ are added to $\mathcal{C}$. The iterations continue until the runtime reaches B. When B is reached, the CRF unary features $\boldsymbol{\psi}_u(\boldsymbol{x}_i, y_i)$ are computed for all supervoxels i where at least one descriptor has been computed. For a supervoxel i, where no descriptors are computed, $\boldsymbol{\psi}_u(\boldsymbol{x}_i, y_i)$ is estimated based on the available unary features $\{\boldsymbol{\psi}_u(\boldsymbol{x}_j, y_j)\}$ of i's neighbors, i.e. $\boldsymbol{\psi}_u(\boldsymbol{x}_i, y_i)$ is set to a weighted sum of $\{\boldsymbol{\psi}_u(\boldsymbol{x}_j, y_j)\}$, where the weights are the inverse Euclidean distances between the centroids of i and the neighbors.

Algorithm 3 Our cost-sensitive budget-aware semantic segmentation inference

```
1:  Input:Supervoxels $V$, Policy $\pi$, Classifier $H$, Budget $B$
2:  Output:Labeling of supervoxels
3:  $\mathcal{C} \leftarrow$ small random subset of $V$;
4:  $\mathcal{F} \leftarrow \emptyset$
5:  repeat
6:      $i \leftarrow$ Select($\mathcal{C}$) % select a supervoxel (see text)
7:      $a \leftarrow \pi(i)$ % apply policy on $i$
8:      if $a ==$ FINISHED then
9:          $\mathcal{F} = \mathcal{F} + \{i\}$
10:         $\mathcal{C} = \mathcal{C} +$ Neighbors($i$) $- \mathcal{F}$ % new candidates
11:     else
12:         Compute the descriptor specified by $a$ for $i$
13:     end if
14: until runtime $< B$
15: Interpolate unary features for supervoxels $i$ with no computed descriptors (see text)
16: Apply CRF inference
```

The key element in the above framework is the policy π. Given a supervoxel i, π must weigh the cost of computing a new descriptor of i versus the potential improvement in accuracy of the CRF inference. It is critical that π makes these decisions efficiently, otherwise the cost of evaluating π would negate any potential reduction in descriptor computation that it provides. As our key contributions, in the following Sec. 5.4, we specify a suitable representation and learning algorithm for such a π.

5.4 Policy Representation and Learning

This section first describes the linear representation we use for policies, and then formulates our policy learning algorithm within the MDP framework.

5.4.1 Policy Representation

At each iteration of our budgeted inference, π is shown a supervoxel i, and asked to decide whether or not to run a new descriptor for i and if so which one. We call these choices actions. π selects an action a based on the information available at the time, which we call the *inference state* s, $a = \pi(s)$. An inference state is a tuple $s = (i, b, \mathcal{C}, \mathcal{F}, \mathcal{D})$, where i is the supervoxel currently being considered, b is the remaining budget, and $\mathcal{C}$ and $\mathcal{F}$ are the sets of candidate and finished supervoxels as described in Sec. 5.3.2. Finally, $\mathcal{D}$ is the set of descriptor outputs that have been produced so far for supervoxels in $\mathcal{C}$ and $\mathcal{F}$.

Since π is called many times during inference, it is critical that the time required to select an action be significantly smaller than the time required to run descriptors. To support efficiency we represent π as a linear function that ranks the possible actions at an inference state s based on an easy to compute vector of π-features $\boldsymbol{\phi}(s)$.

Policy Features. The π-features have three subvectors $\boldsymbol{\phi}(s) = [\boldsymbol{\phi}_1(s), \boldsymbol{\phi}_2(s), \boldsymbol{\phi}_3(s)]$ that capture different aspects of the inference state s. $\boldsymbol{\phi}_1(s)$ is a binary vector that indicates which descriptors have already been run for i. $\boldsymbol{\phi}_1(s)$ allows the policy to learn the value of taking certain actions, given various combinations of computed descriptors characterizing s. $\boldsymbol{\phi}_2(s)$ is a vector that is equal to a weighted average of the unary potential features $\{\boldsymbol{\psi}_u\}$ of "finished" neighbors of i that are in $\mathcal{F}$. Recall that $\boldsymbol{\psi}_u$ corresponds to probability distributions over class labels. Thus, $\boldsymbol{\phi}_2(s)$ allows the policy to base its decisions on the confidence of neighboring supervoxels about the various semantic labels. For example, the policy can learn that if all neighbors are very confident about a particular label then it is not worth computing further descriptors for i. Finally, $\boldsymbol{\phi}_3(s)$ is the standard shape-context descriptor capturing the spatiotemporal layout of finished supervoxels in $\mathcal{F}$ around i. $\boldsymbol{\phi}_3(s)$ is computed by binning the space-time neighborhood of i (all supervoxels that touch i) into 8 bins ($\{up, down, left, right\} \times \{before, after\}$), and counting the finished supervoxels that fall in each bin. $\boldsymbol{\phi}_3(s)$ allows the policy to base its decisions in part on the density of surrounding finished supervoxels.

Given the π features for an inference state s, $\boldsymbol{\phi}(s)$, our policy is a linear ranking function over policy actions a, represented in terms of a weight vector $\boldsymbol{w}_a$ for each action.

$$\pi(s) = \arg\max_a \ \boldsymbol{w}_a \cdot \boldsymbol{\phi}(s) \tag{5.2}$$

Thus, training the policy entails training the weights $\boldsymbol{w}_a$ so as to make the best possible decisions.

5.4.2 Policy Learning

It is important to note that ground-truth annotations of our training videos do not directly provide ground-truth decisions that should be made by π. The supervised training data for π would need to label inference states by best policy actions. Since this information is not available in training, π cannot be learned via pure supervised learning. The training data does, however, provide the means for evaluating the quality of any policy. In particular, given any π and budget B, we can run time-bounded inference on each training video using π, as summarized in Alg. 3, and then measure the prediction accuracy of the CRF relative to the available ground truth. In practice, large numbers of such policy evaluations can be run quickly by precomputing all descriptors for all supervoxels across the training videos. This allows for the budgeted inference process to be "simulated" on the training data without requiring descriptors to be recomputed for each policy evaluation. The question then is how to use this fast policy evaluation on the training data in order to learn an effective policy?

Policy learning is complicated by the fact that the policy is inherently solving a sequential decision making problem, where each decision may have long-term impacts on the overall solution accuracy. Optimizing the long-term value of policies is challenging due to the fact that each inference process will involve many policy decisions and assigning relative credit to those decisions toward the overall accuracy is non-trivial. Such sequential decision making problems are naturally formalized in the MDP framework [122].

MDP Formulation: An MDP specifies a set of states S, a set of actions A that can be taken by a policy, and a transition function T, which describes how the state of the system changes when actions are taken. In addition, an MDP specifies a reward function, which evaluates the relative goodness of various system states. In our time-bounded inference application, the states correspond to inference states $(i, b, \mathcal{C}, \mathcal{F}, \mathcal{D})$ as described in Sec. 5.4.1. The actions A correspond to policy actions of either selecting to run a descriptor for the current supervoxel i, or returning FINISHED, which indicates that we are finished computing descriptors for i.

The transition function describes how an action a changes a state $s = (i, b, \mathcal{C}, \mathcal{F}, \mathcal{D})$. If the action a is to run a descriptor, then the new state is equal to $s' = (i', b', \mathcal{C}, \mathcal{F}, \mathcal{D}')$, where $\mathcal{D}'$ updates $\mathcal{D}$ with the newly computed descriptor information for i. In addition, the new budget b' will be equal to b minus the cost of a, and $i' \in \mathcal{C}$ will be the supervoxel selected next for processing. When the action is FINISHED, the new state is $s' = (i', b, \mathcal{C}', \mathcal{F}', \mathcal{D})$, where $\mathcal{C}'$ and $\mathcal{F}'$ are updated as described in Sec. 5.3.2, and i' is the newly selected supervoxel. Note that when $b = 0$, no further actions are allowed. Importantly, the reward function is zero for all states, except for final states with $b = 0$, where the reward is equal to the accuracy achieved by the CRF inference using the selected descriptors run by the π.

Given the above MDP formulation, the problem of optimizing π to maximize expected long-term reward in the MDP is identical to finding a policy that maximizes inference accuracy within our budgeted inference framework. Thus, in principle, any policy learning algorithm from the MDP literature could be employed for our problems. Prior work [163] used reinforcement learning (RL) for learning an approximate Q-function. In that work, the Q-function $Q(s, a)$ of an MDP gives the expected future reward of being in state s and taking action a. Given $Q(s, a)$ the policy is defined to select the action with largest Q-value. Unfortunately, the Q-function can be extremely complicated to represent for problems that involve long sequences of decisions. In that prior work, the number of decisions was bounded by the number of descriptors, which is quite small. Rather, in this paper, the number of decisions is related to the number of supervoxels, which is substantially larger. Our early experiments showed that standard approaches for learning Q-functions, represented using our π-features, were ineffective for the problem scales we address here.

Classication-Based Approximate Policy Iteration (CAPI): Our approach is motivated by the fact that we do not actually need to learn the Q-function, but only learn to rank good actions above bad actions, e.g., using the linear policy representation described in Sec. 5.4.1. This suggests considering approaches that directly learn the decision boundary between good and bad actions. Classification-Based Approximate Policy Iteration (CAPI) is one such state-of-the-art technique that we follow here.

CAPI was originally proposed by [34, 83]. It has demonstrated a number of empirical successes, and has been the subject of theoretical analysis providing various performance guarantees [35, 23, 88]. A key distinction of CAPI is that it is able to leverage state-of-the-art learning algorithms for classification and ranking (e.g. SVMs).

CAPI is conceptually simple. Given an initial policy π_0, which is random in our experiments, CAPI iteratively applies an *approximate policy improvement operator* $\mathcal{PI}$, which takes an input policy π and returns an (approximately) improved policy $\mathcal{PI}[\pi]$. Thus, the CAPI algorithm produces a sequence of improving policies $\pi_{t+1} = \mathcal{PI}(\pi_t)$ and terminates when no further improvement is observed or a training time bound is reached. Recall that for our linear policy representation this will correspond to a sequence of weight vectors. It remains to describe the approximate policy improvement operator $\mathcal{PI}$.

Given a current policy π, CAPI computes the improved policy $\mathcal{PI}(\pi)$ using a two step process:

- **Step 1 – Training Set Generation.**Create a training set of state-action pairs Trn $= \{(s_i, a_i)\}$ such that a_i is an "improved" action for s_i, i.e. better or at least as good as the current action $\pi(s_i)$.
- **Step 2 – Classifier Learning.** Apply a classifier learner to Trn to obtain a policy π' that achieves high accuracy in selecting the improved actions. In Step 2 we use a multi-class SVM classifier to learn the weights of π based on the training set generated by Step 1.

Step 1 requires selecting a set of states for the training set and then computing the improved actions. In our experiments, we have found that an effective and simple approach is to run the current policy π on the training videos, and to let the training states correspond to all inference states encountered during inference using π across all training videos. For each such inference state s_i we must now compute a label a_i that corresponds to an improved action relative to the action selected by π.

In order to compute an improved action, CAPI uses the Monte Carlo simulation technique of *policy rollout* [145]. Fig. 5.2 illustrates the main components of policy rollout. Simply stated, policy rollout computes a score for each action a at a state s_i that is equal to the accuracy achieved by the final CRF inference after taking action a in s_i and then taking actions according to π until the budget is zero. The action leading to the highest accuracy is then selected. That is, rollout considers all one-step action departures from the current policy π at s_i and selects the action that resulted in the highest final CRF accuracy. Note that the transition function for actions may be stochastic, e.g., when the selection of the next supervoxel is implemented by random selection. In these cases, policy rollout runs multiple simulations for each action and the average accuracies across simulations are used to score actions.

For deterministic transition functions, policy rollout is guaranteed to return an improved action if the current policy is not optimal in s_i. For stochastic transitions, in the limit of infinite simulations, the action returned by rollout is guaranteed to be an improved action. A polynomial sample complexity bound exists on the quality of the action returned by rollout compared to that of π [35]. Since our classifier is linear the learning time is linear in sample size. Note that while the training process may require significant time, the end result is a single policy π from the last iteration of CAPI that can be used efficiently at test time for time-bounded inference.

5.5 Results

Our goal is to empirically support the claim that we can optimally adapt a given method for semantic video segmentation to varying time budgets, such that it yields satisfactory performance for any budget, and maintains an accuracy as close as possible to its performance for no time bound. As the given method was originally designed to perform best without time constraints, it is important to note that our performance is inherently upper-bounded by the method's accuracy for an infinite time budget. Therefore, our evaluation differs from much work in computer vision, where the focus is on demonstrating improvements in accuracy.

Datasets. For evaluation, we use three benchmark datasets: 1) CamVid [12], 2) MPIScene [169], and 3) SUNY Buffalo-Xiph.org 24-class [15]. CamVid consists of 5 videos with an average length of 5000 frames. The videos are captured with a moving camera recording road scenes. Following prior work, we focus on the 11 most common object class labels and use the standard split of training and test frames as in [12, 96]. MPIScene consists of 4 videos with an average length of 150 frames. The videos show driving scenes recorded from a car. Almost 25% of the frames are labeled with 5 object classes. For MPIScene, we use the split of 1/2 training and 1/2 test frames as in [170]. The SUNY Buffalo dataset consists of 10 videos of diverse scenes with an average length of 80 frames. They are fully annotated with 24 class labels. As in [61], we use one-half of the frames for training and the other half for testing.

Supervoxels and Descriptors. All videos are partitioned into supervoxels using the hierarchical graph-based approach of [51], and its software implementation presented in [176]. For

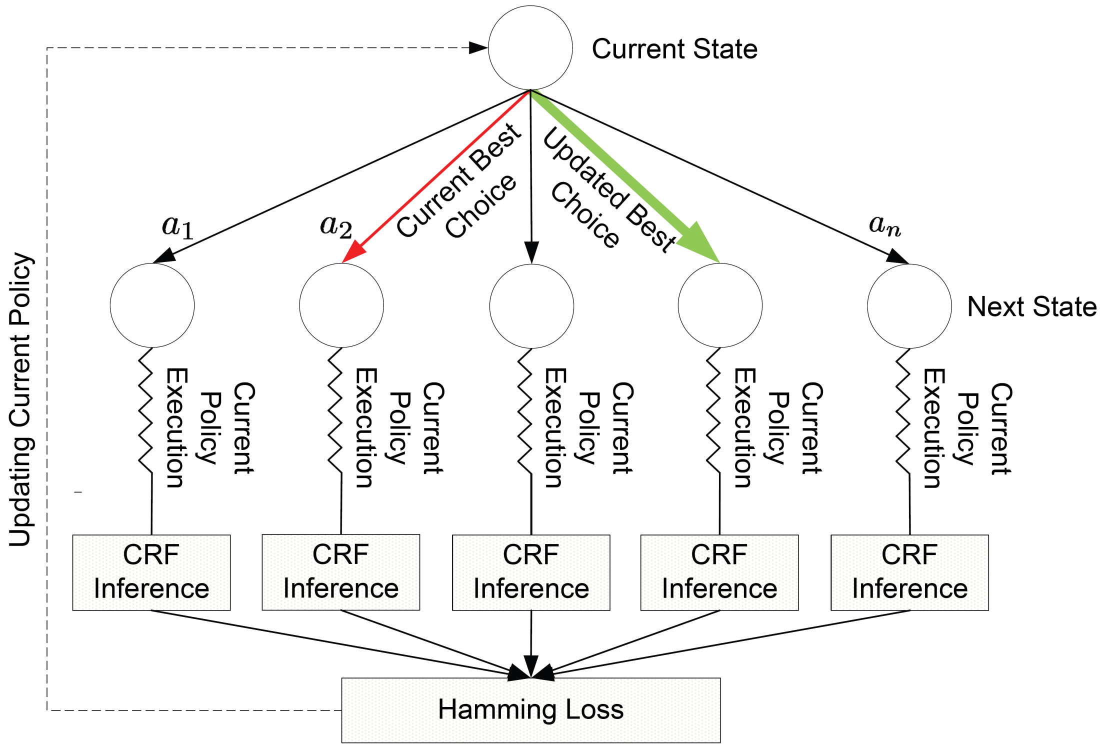


Figure 5.2: Monto Carlo simulation of policy rollout. All possible actions of a current state are evaluated by running the current policy π starting from the next states corresponding to the actions being taken, until the time budget is reached. The CRF inference is then applied to compute the Hamming loss. This is used to improve the current policy.

CamVid, the video partitioning is based on the 9th level of the generated supervoxel tree, producing on average 3500 supervoxels per video. For, MPIScene, we used the 9th level of the supervoxel hierarchy, producing on average 1000 supervoxels per video. In this paper, we do not consider the computation time of extracting supervoxels.

Each supervoxel has access to algorithms for computing appearance and motion descriptors, as controlled by our inference policy. These descriptors include the following. We use dense trajectories [153] by tracking a set of densely sampled points in two different scales on a grid. The descriptor extraction finds the tightest cube of a given supervoxel, and uses the dense trajectories to generate HOG, HOF, and MBH (motion boundary histogram) for each track. We also use the color histogram in CIE-Lab color space for each supervoxel. In addition, we also use object detectors as mid-level descriptors of supervoxels to identify the probability of observing a corresponding object class in a supervoxel. Specifically, given a supervoxel, we run a Deep Convolutional Neural Net (DCNN) on all pixels of the supervoxel's first, middle and last frame. Then, the output of the logistic regression layer of DCNN is used as a descriptor of the superpixel. For CamVid, computing HOG, HOF and MBH for the entire video takes 6-8, 15-17, 9-11 minutes, respectively; computing the color histogram for all supervoxels in a video takes 3-5 minutes; and running DCNN takes 0.1 seconds per supervoxel. On average the full descriptor extraction for an entire video in CamVid requires 43-53 minutes. For MPIScene, computing HOG, HOF and MBH for an entire video takes 10-12, 18-20, 11-12 seconds, respectively; computing the color histogram for all supervoxels in a video takes 2-4 seconds; and running DCNN takes 0.1 seconds per supervoxel. On average the full feature extraction for an entire video in MPIScene requires 42-50 seconds. For SUNY Buffalo-Xiph.org 24-class computing HOG, HOF and MBH for the entire video takes 5-7, 8-10, 5-6 seconds respectively. Computing the color histogram over all supervoxels in a video takes 0.6-1 second. On average, full feature extraction for the entire video from the SUNY Buffalo dataset requires 19-22 seconds.

Variations of Our Framework: We evaluate our framework using different supervoxel selection strategies (Sec. 5.3.2) and different sets of available descriptors. We consider three variations: 1) *Budget -RndRnk* randomly selects a supervoxel from $\mathcal{C}$, and uses only HOG, HOF, MBH and color histogram; 2) *Budget -NhbRnk* ranks the supervoxels based on the confidence of H classifiers for neighbors in $\mathcal{F}$ (Sec. 5.3.2), and uses only HOG, HOF, MBH and color histogram; 3) *Budget -Full* is similar to Budget -NhbRnk, but additionally uses DCNN-based descriptor.

Upper-Bound Performance: For evaluating our upper-bound performance, we compare our time-bounded accuracies to two unbounded CRF models. The first model, referred to as CRF, uses only HOG, HOF, MBH and color histogram descriptors. The second model, referred to as CRF-Full, additionally uses DCNN-based descriptor.

Baselines: We specify a number of reasonable baseline approaches. This comparison serves to evaluate how our proposed learning of the inference policy affects performance relative to alternative strategies. 1) *Baseline1* randomly selects a sequence of descriptors to be computed for all supervoxels until time runs out; 2) *Baseline2* randomly selects a subset of supervoxels, such that there is time for computing all descriptors for each supervoxel in the subset, and 3) *Baseline3* is aimed to recreate the related approach of [163] in our domain, i.e., *Baseline3* learns an inference policy using Q-learning for selecting a sparse set of descriptors that are computed for all supervoxels in the entire video.

Implementation: is done in C++. Darwin library(http://drwn.anu.edu.au/) is used for training the CRF and H. The α-expansion algorithm [11] is used for CRF inference. We perform our experiments on an Intel quad core-i7 CPU and 16GB RAM PC. We use Caffe deep learning framework [64] for our DCNN implementation. Fine tuning of the DCNN parameters based on the Alexnet model is done for each dataset. The training set for the objects is obtained as in [96].

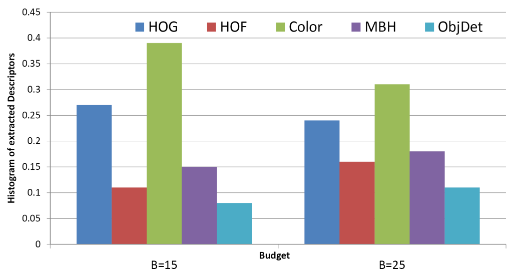
Table 5.1 compares per-class and average video labeling accuracy of the aforementioned variations of our framework with those of time-unconstrained CRF models and baselines, for three different time budgets, on CamVid. As can be seen, for the smallest budget, the accuracy of all baselines is much worse than that of all our framework variations. We observed that Baseline1 and Baseline2 were not able to use HOF and MBH, for this budget, because the cost of computing motion descriptors for the entire video was above the budget. This demonstrates the importance of our intelligent descriptor selection.

In Table 5.1, we also see that all variations of our framework are able to continually improve accuracy as the time budget increases. At the largest time budget of 45 minutes, Budget -Full achieves nearly the same accuracy as the unbounded CRF-Full which in turn requires 50-55min for computation of all descriptors in the entire video. This demonstrates that we are able to maintain a similar level of accuracy of the original method under reduced runtimes, namely for a 5 min time reduction.

Interestingly, the results in Table 5.1 show that for lower budgets the variations of our framework give superior accuracy for the dominant class labels relative to unbounded inference. This is likely due to our explicit capturing of the contextual information from neighboring supervoxels. The results suggest that our policy successfully learned to avoid computing redundant

B	Method	Road	Bldg	Sky	Tree	SWlk	Car	Pole	Fence	Pdstr	Bcyl	Sign	Avg
∞	CRF	90.2	74.2	95.2	79.8	69.8	75.8	10.1	29.2	59.9	35.4	50.2	60.9
	CRF-Full	90.5	74.6	95.2	80.1	70.3	78.8	10.4	30.1	59.4	37.2	50.4	61.5
10	Baseline1	81.1	65.3	74.2	39.9	30.2	46.3	3.9	7.2	19.5	10.3	17.1	35.9
	Baseline2	91.9	70.9	84.4	51.6	36.2	53.2	5.8	10.4	26.2	14.1	27.2	42.9
	Baseline3	89.1	71.3	83.2	47.9	33.8	50.4	6.2	11.1	25.8	13.4	24.5	41.5
	Budget -RndRnk	93.2	75.6	90.3	69.4	51.3	58.8	6.2	12.2	27.2	13.2	23.2	47.3
	Budget -NhbRnk	91.2	76.4	91.5	71.2	50.6	56.9	7.4	13.5	28.1	15.4	25.7	48.0
	Budget -Full	91.9	78.9	94.2	73.4	53.8	62.4	8.1	14.1	36.6	24.5	28.8	51.5
25	Baseline1	86.9	73.8	79.8	50.7	49.1	52.2	7.2	13.6	30.2	21.5	26.4	44.7
	Baseline2	89.3	75.9	88.2	68.8	40.5	60.7	7.5	15.4	38.5	25.3	29.2	49.0
	Baseline3	89.4	72.3	89.2	69.1	35.1	57.2	6.8	13.9	33.7	20.1	25.3	46.6
	Budget -RndRnk	90.6	78.9	93.4	75.1	52.9	65.5	6.7	13.5	32.6	20.7	29.1	50.8
	Budget -NhbRnk	92.9	77.7	93.2	76.6	57.4	67.5	8.1	16.7	37.4	24.5	28.9	52.8
	Budget -Full	92.7	77.4	96.9	79.1	63.3	73.1	9.7	20.7	44.2	29.8	36.2	56.6
45	Baseline1	88.4	72.8	92.3	78.8	68.7	76.5	10.0	24.4	57.9	35.8	48.1	59.4
	Baseline2	87.5	73.2	91.7	73.5	65.2	75.4	9.9	23.7	51.6	32.8	47.8	57.5
	Baseline3	89.6	70.9	91.2	77.5	66.7	73.9	10.4	25.8	57.8	33.9	50.5	58.9
	Budget -RndRnk	89.8	66.2	92.7	77.3	64.3	72.5	9.1	24.6	53.8	32.4	47.1	57.3
	Budget -NhbRnk	90.3	72.9	89.8	76.9	64.8	76.7	9.3	27.8	56.9	34.4	46.5	58.8
	Budget -Full	90.8	74.5	91.7	79.4	68.1	75.0	10.2	28.3	58.3	38.2	49.9	60.4

Table 5.1: Per-class and average accuracy on CamVid. We evaluate CamVid for $B \in \{10, 25, 45$ minutes$\}$. The upper-bound accuracy for Budget -RndRnk and Budget -NhbRnk is the accuracy of the CRF and the upper-bound accuracy for Budget -Full is the accuracy for CRF-Full. The following abbreviations are used: Bldg = Building, SWalk = Side Walk, Pole = Column-Pole, Pdstr = Pedestrian, Bcyl = Bicycle, Sign = Sign Symbol



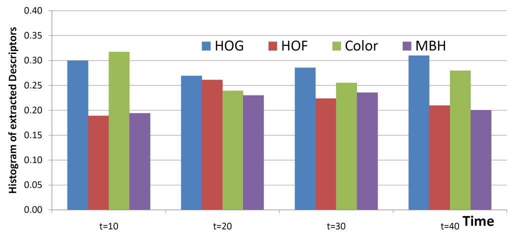


Figure 5.3: (top) Histogram of descriptors selected by *Budget -Full* for different budgets on CamVid. (bottom) Histogram of descriptor selection by *Budget -NhbRnk* in time on CamVid, constrained by $B = 45min$.

descriptors when the neighbors can provide strong evidence of the label. Thus, the main impact of increasing the budget is to improve accuracy on the non-dominant labels.

Tables 5.2 show the per-class and average video labeling accuracy on MPI-Scene and SUNY Buffalo-Xiph.org 24-class dataset.

Fig. 5.3(top) how decisions of our policy, learned in *Budget -Full*, differ across various budgets for CamVid. Specifically, the figure shows the histogram of certain types of descriptors selected for different budgets. We see that HOF and MBH are seldom used for small budgets, as expected, since they incur higher costs. Also, the color histogram descriptor is more frequently selected when we are given small budgets, as expected, since they incur small costs.

Fig. 5.3(bottom) shows how the distribution of descriptor selection changes in time during our budgeted inference by *Budget -NhbRnk* for CamVid. Specifically, the figure shows the histogram of descriptor selections made by the policy at various moments in time of the inference process. We see that the distribution changes as the inference time increases until the budget is reached. For example, initially, the distribution is highly skewed toward selections of the cheap color descriptor but then becomes more uniform. This suggests that the policy successfully learned to select low-cost descriptors initially for facilitating later policy decisions.

5.6 Summary

We formulated the new problem of budgeted semantic video segmentation, where the pixels of a video must be semantically labeled under a time budget. We presented a budgeted inference framework for this problem that intelligently selects supervoxel descriptors to run, which are

B	Method	Bkgd	Road	Lane	Vehicle	Sky	Avg
∞	CRF	87.3	91.2	11.5	66.2	94.5	70.1
	CRF-Full	89.8	90.4	12.1	69.8	94.9	71.4
15	Baseline1	60.3	80.4	3.6	31.4	82.4	51.6
	Baseline2	63.7	81.9	3.5	32.7	83.7	53.1
	Baseline3	65.9	81.8	3.9	31.9	83.5	53.4
	Budget -RndRnk	70.3	82.9	7.3	39.2	84.1	56.8
	Budget -NhbRnk	73.5	85.2	8.9	42.4	86.7	59.3
	Budget -Full	74.6	85.6	9.1	44.8	87.2	60.3
30	Baseline1	74.8	84.2	6.9	36.4	85.5	57.6
	Baseline2	75.9	87.3	6.5	39.3	84.9	58.8
	Baseline3	75.0	85.7	7.2	38.9	83.5	58.1
	Budget -RndRnk	80.1	83.4	9.4	49.7	88.7	62.3
	Budget -NhbRnk	83.6	86.8	9.8	52.9	89.3	64.5
	Budget -Full	84.1	88.4	10.1	54.9	89.4	65.4
45	Baseline1	88.3	91.4	10.9	66.4	93.2	70.0
	Baseline2	86.8	90.1	9.3	63.9	94.1	68.8
	Baseline3	89.2	91.3	11.1	66.6	94.7	70.6
	Budget -RndRnk	86.8	90.7	10.1	64.8	93.9	69.3
	Budget -NhbRnk	88.9	91.2	10.4	65.6	94.7	70.2
	Budget -Full	89.8	91.5	11.1	70.9	94.9	71.6

(a) Results on MPI-Scene

B	Method	Acc
∞	CRF	52.1
10	Baseline1	32.2
	Baseline3	32.8
	Budget -RndRnk	38.7
	Budget -NhbRnk	39.6
15	Baseline1	37.4
	Baseline3	37.9
	Budget -RndRnk	44.7
	Budget -NhbRnk	46.6
20	Baseline1	49.3
	Baseline3	49.8
	Budget -RndRnk	48.7
	Budget -NhbRnk	49.5

(b) Results on SUNY Buffalo

Table 5.2: Per-class and average accuracy on MPI-Scene (Left) and SUNY Buffalo (Right). We evaluate MPI-Scene for $B \in \{15, 30, 45 \text{ seconds}\}$ and SUNY Buffalo for $B \in \{10\text{sec}, 15\text{sec}, 20\text{sec}\}$. The upper-bound accuracy for Budget -RndRnk and Budget -NhbRnk is the accuracy of the CRF and the upper-bound accuracy for Budget -Full is the accuracy for CRF-Full. The following abbreviation is used: Bkgd = Background

then used for CRF inference. Since descriptor computation often dominates the cost of CRF inference, our framework can provide substantial time savings in a principled manner. We formulated the inference policy for selecting among descriptors to run for each supervoxel in a video. We introduced a principled algorithm for learning such policies based on labeled training videos by formulating budgeted inference in the framework of an MDP. Our experiments show that we are able to learn policies for budgeted inference that significantly improve on the accuracies of several baselines. The results also demonstrate that we can optimally adapt a method, design to operate with no time bound, to varying time budgets, such that it yields satisfactory performance for any budget, and maintains an accuracy as close as possible to its performance for no time bound.

Chapter 6: Budget-Aware Deep Semantic Video Segmentation

Abstract

In this work, we study a poorly understood trade-off between accuracy and runtime costs for deep semantic video segmentation. While recent work has demonstrated advantages of learning to speed-up deep activity detection, it is not clear if similar advantages will hold for our very different segmentation loss function, which is defined over individual pixels across the frames. In deep video segmentation, the most time consuming step represents the application of a CNN to every frame for assigning class labels to every pixel, typically taking 6-9 times of the video footage. This motivates our new budget-aware framework that learns to optimally select a small subset of frames for pixelwise labeling by a CNN, and then efficiently interpolates the obtained segmentations to yet unprocessed frames. This interpolation may use either a simple optical-flow guided mapping of pixel labels, or another significantly less complex and thus faster CNN. We formalize the frame selection as a Markov Decision Process, and specify a Long Short-Term Memory (LSTM) network to model a policy for selecting the frames. For training the LSTM, we develop a policy-gradient reinforcement-learning approach for approximating the gradient of our non-decomposable and non-differentiable objective. Evaluation on two benchmark video datasets show that our new framework is able to significantly reduce computation time, and maintain competitive video segmentation accuracy under varying budgets.

6.1 Introduction

We consider the problem of semantic video segmentation, where the goal is to assign a correct class label to every pixel in a video. Recently deep networks have achieved state-of-the-art results for semantic video segmentation [102, 113, 73, 182, 18], where typically a Convolutional Neural Network (CNN) is applied to directly label each pixel of each frame. In this way, they significantly improve on the the processing time of more traditional approaches (e.g., energy minimization), as the latter requires time for graphical-model inference and pre-processing time for feature extraction. However, despite the feed-forward architecture of CNNs, and their parallelizable computation on GPUs, runtimes are still far from real time. For example, it takes 6-9 times the video length to execute the approach of [9] on benchmark datasets. Unfortunately, this rules out the practical use of these approaches for applications with tighter runtime constraints. In some situations hardware solutions can help meet the constraints. However, it is equally important to develop a better understanding of how to achieve maximal accuracy given constraints on the compute time and resources.

In this paper, we address the above problem by introducing a framework for budget-aware deep semantic video segmentation. Our approach is applicable to any available semantic segmentation network, which is treated as a black box. Given a semantic segmentation network and a time budget, our approach attempts to maximize accuracy within the budget. The main idea is based on the observation that videos typically show smooth motions, and hence pixel labels of a frame can be efficiently and accurately interpolated from neighboring segmentations. This motivates intelligently selecting frames to be processed by a deep segmentation network and then using fast interpolation networks to assign pixel values to unselected frames. When the interpolation network is significantly faster than the segmentation network there can be significant computational savings.

The problem of budget aware inference is receiving increasing attention as state-of-the-art accuracies approach the needs of many applications. Recently, this problem has been studied for activity detection in video [179], where the goal is to efficiently identify activities using deep networks while avoiding the need to process all video frames. In that work, a recurrent "attention model" was learned which aimed to select a small subset of the video frames for processing, while maintaining high accuracy. The results showed that high accuracies could be maintained while only processing approximately 2% of the video frames. While this result suggest that intelligent frame selection is a viable way to speedup deep architectures, the problem of activity detection is quite different from our problem of semantic segmentation. In particular, unlike activity detection, the loss function for semantic segmentation is defined over all pixels of a video and it is necessary to assign predicted values to all pixels. This raises the question of what accuracy-time tradeoffs can be achieved for semantic segmentation.

Three Key Ideas: We extend the state of the art by developing: (1) A deep visual attention model, represented as an LSTM network, aimed at optimally selecting only a subset of video frames whose total time of processing by a segmentation network would not exceed a budget constraint. (2) A fast interpolation model, represented by a one-layer CNN, for efficiently labeling the remaining unprocessed frames based on neighboring frames selected for segmentation by the attention model; and (3) Joint learning of main components of our approach – namely, the attention and interpolation models, such that accuracy of the resulting video segmentation is maximized for a given budget.

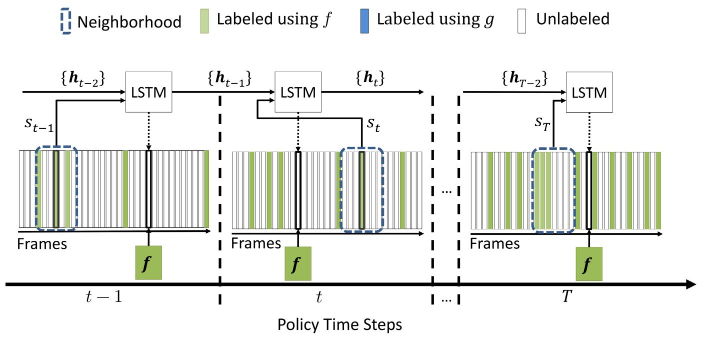
Fig. 6.1 illustrates the two stages of our approach. In Stage 1 (Fig. 6.1 left), the LSTM policy is run to select T frames for which to apply the segmentation network f. In Stage 2 (Fig. 6.1 right), the interpolation model $\boldsymbol{g}$ is applied to the remaining video frames. As shown in Sec. 6.4, the total execution time of our approach is a strictly increasing function of T. Hence, from the constraint that our total runtime should be less than the budget B, we can easily estimate the optimal T, by simply stopping the LSTM policy before we exceed the budget.

Our joint training approach is based on recognizing that our problem is inherently one of sequential decision making. Accordingly we draw on ideas from reinforcement learning, which is an area that focuses on learning for sequential decision making. In particular, we derive a policy-gradient reinforcement learning algorithm for our problem, which is shown to be an effective approach for training our models.

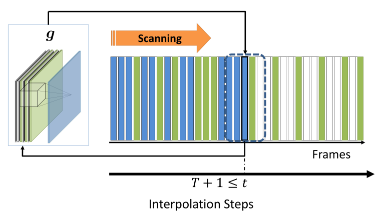
In Sec. 6.5 we evaluate our approach for varying time budgets and two different semantic segmentation netoworks. Our results show that for budgets $\frac{1}{4}B_{\text{max}} \leq B \leq \frac{1}{2}B_{\text{max}}$, we achieve semantic segmentation accuracy that is comparable to the results obtained by directly segmenting each frame. This shows that our approach is able to learn to speedup the segmentation performance by a factor of four, with little loss in accuracy. Moreover, our accuracy gracefully degrades for $B < \frac{1}{4}B_{\text{max}}$, which is important for applications with very tight budget constraints.

6.2 Related Work

Semantic video segmentation is a long standing problem, and a thorough review of the literature is beyond our scope. It has been traditionally formulated as an energy minimization of a graphical model representing supervoxels or superpixels in the video [15, 13, 12, 107, 16, 96, 143, 170, 100, 67]. Except for a few empirical results of sensitivity to the total number of supervoxels [61, 95] or greedy feature selection in [50], these approaches usually do not explicitly study trade-offs between accuracy and efficiency under varying time constraints. Our main hypothesis is that knowing the budget constraint in training provides additional information which allows the learning algorithm to optimize its decisions toward maximizing overall labeling accuracy. More importantly, it is hard to adapt the approaches proposed in [61, 95, 50] for deep learning base semantic segmentation models.



Stage 1: Running inference policy for T steps.



Stage 2: Predicted pixel labels are propagated to remaining $M - T$ unlabeled frames.

Figure 6.1: Two stages of our approach. Given a video with M frames and time budget B, in Stage 1, LSTM-based policy sequentially selects a subset of frames for segmentation by a CNN, $\boldsymbol{f}$. In Stage 2, the missing pixel labels are interpolated by a one-layer convolution filter, $\boldsymbol{g}$, using neighboring semantic segmentations.

Recent work on CNN-based semantic segmentation [9, 73, 102, 57, 113, 191, 17, 93, 116, 132] does not require unsupervised segmentation in a pre-processing step, but directly take pixels of the image as input and output a semantic segmentation. These approaches first use a set of (convolution + pooling) layers to generate a deep feature for the entire image. Then, they perform a sequence of deconvolution + upsampling operations for generating an output feature map. As these approaches [9, 73, 102, 57, 113, 191, 17, 93, 116, 132] conduct CNN-based segmentation for every frame independently, their pixel labeling is typically *not* spatiotemporally smooth and coherent. Recently, Peng el al. [90] propose a recurrent temporal field model which considers smoothness of labels is space-time. Also, the runtime of these feed-forward CNN architectures is typically 6-9 times the video length [113].

Efficient inference under budget has recently received much traction in many areas of computer vision [188, 80, 11, 4, 69, 70, 172, 162, 163, 129, 94, 50]. These approaches typically model a utility function of their inference steps, and economically run those steps with the maximum utility. Our key difference is in that we directly learn a budget aware inference policy that achieves high utility within the deep learning framework. These approaches are not easy to generalize to the state-of-the-art CNN based semantic segmentation, since such approaches do not explicitly extract features during a preprocessing step. Unlike these approaches we do not require extracting additional features for estimating the utility, but instead train our deep budget-aware semantic segmentation model in an end-to-end fashion. Vijayanarasimhan et al. [148] use the idea of informative frame selection and label propagation to facilitate human annotation in semantic video segmentation in an active learning framework. Their work is different than ours in: 1) They approach is built on top of the traditional CRF based semantic labeling, 2) Their full model requires solving another CRF model which takes at least a minute itself, and 3) They are trying to improve the human labeling time which is in the order of 25 minutes per frame.

6.3 Problem Formulation

Given a video, $\boldsymbol{x}$, with M frames and budget B, our goal is to accurately assign a label to each pixel in $\boldsymbol{x}$ in time less than B. Let $\boldsymbol{f}$ be a *segmentation function* that takes the pixels $\boldsymbol{x}_i$ of frame i and returns a semantic segmentation of $\boldsymbol{x}_i$ using time $c_{\boldsymbol{f}}$. In particular, $\boldsymbol{f}(\boldsymbol{x}_i)$ gives a posterior distribution over class labels for each individual pixel. We specify $\boldsymbol{f}$ as a CNN following two prior approaches [9, 73]. To meet the budget constraint, we apply $\boldsymbol{f}$ to only a subset of frames and interpolating the resulting segmentations to other frames. For this we use a *visual attention policy*, π, which sequentially selects frames for labeling by $\boldsymbol{f}$ (details in Section 6.4.1). Given the output of $\boldsymbol{f}$ at the selected frames, remaining frames are labeled by an *interpolation function*, $\boldsymbol{g}$, which uses nearby outputs of $\boldsymbol{f}$ to interpolate semantic segmentations to yet unprocessed frames. Importantly, the time cost of applying $\boldsymbol{g}$, $c_{\boldsymbol{g}}$, is significantly smaller than the time cost of applying $\boldsymbol{f}$, $c_{\boldsymbol{f}}$, which allows for time savings compared to labeling all frames via $\boldsymbol{f}$.

Since the goal is to produce a segmentation within a time budget B, we must decide when to stop selecting frames with π in order to meet the budget constraint. For this purpose, we can divide the total time required to compute an output into two components. The first time component is the time required to apply the policy π for T steps and also apply $\boldsymbol{f}$ at the selected frames. Let $\boldsymbol{u}^{(T)} \in \{0,1\}^M$ denote an indicator vector over video frames, where $u_i^{(T)} = 1$ means that frame i has been segmented by π, and $u_i^{(T)} = 0$, otherwise. Also let $U^{(T)} = |\boldsymbol{u}^{(T)}| \leq T$ be the number of distinct frames selected by π, noting that this can be less that T if π happens to select a frame twice.[1] The second time component involves applying $\boldsymbol{g}$ to frames with zero values in $\boldsymbol{u}^{(T)}$. If we let c_π denote the cost of applying the policy, then the total runtime is given

[1]While π will rarely reselect a frame, we cannot rule out this possibility since π is learned and will have some imperfections.

by:

$$\begin{aligned} C(T) =& c_\pi T + c_{\boldsymbol{f}} U^{(T)} + c_{\boldsymbol{g}}(M - U^{(T)}) \\ =& c_\pi T + (c_{\boldsymbol{f}} - c_{\boldsymbol{g}})U^{(T)} + c_{\boldsymbol{g}} M \end{aligned} \tag{6.1}$$

Using this runtime formula it is easy determine when to stop the policy. After T policy selections, we must stop before running step $T + 1$ if this could result in $C(T + 1) > B$.

In our work, π, $\boldsymbol{g}$, and $\boldsymbol{f}$ will be represented via deep neural networks. Since $\boldsymbol{f}$ is a pre-trained model, model parameters consist of the parameters of π and $\boldsymbol{g}$, i.e. $\theta = (\theta_\pi, \theta_{\boldsymbol{g}}$. Given a video $\boldsymbol{x}$ and time budget B, we let $\hat{\boldsymbol{y}}(\boldsymbol{x}, B, \theta)$ denote the output of the above semantic segmentation process using parameters θ applied to $\boldsymbol{x}$ with budget B. When clear from context we will denote this via $\hat{\boldsymbol{y}}$. Note that $\hat{\boldsymbol{y}}_i$ gives the posterior probability over labels for each pixel in frame i of the video. The loss of a frame labeling will be denoted by $\Delta(\boldsymbol{y}_i, \hat{\boldsymbol{y}}_i)$, where $\boldsymbol{y}_i$ is the ground truth labeling for frame i and Δ is an average cross-entropy loss for pixels in frame i. Further, the loss over M video frames is defines as:

$$L_\theta(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \frac{1}{M} \sum_{i=1}^{M} \Delta(\boldsymbol{y}_i, \hat{\boldsymbol{y}}_i). \tag{6.2}$$

Given a training set of N labeled videos $\{(\boldsymbol{x}^n, \boldsymbol{y}^n\}$ the goal of learning is to find θ that minimizes the following

$$\theta^* = \arg\min_\theta \left[L(\theta) = \mathbb{E}(L_\theta) \approx \frac{1}{N} \sum_{n=1}^{N} L_\theta(\boldsymbol{y}^n, \hat{\boldsymbol{y}}^n) \right] \tag{6.3}$$

Unfortunately, even for simple choices for π, $\boldsymbol{f}$, and $\boldsymbol{g}$, the gradient of $L(\theta)$ does not have a closed form, due to the sequential nature of the process used to construct $\hat{\boldsymbol{y}}^n$.

6.4 Learning and Representation

In this section, we describe our representation for π and $\boldsymbol{g}$, noting that we use existing image segmentation models for $\boldsymbol{f}$, SegNet [9] and BayesianSegNet [73], whose detailed specification can be found in the respective references. We then describe our approach for jointly learning these functions by drawing on policy-gradient estimation techniques.

6.4.1 LSTM-based Policy

Our temporal attention policy makes a sequence of frame selections based on the local information perceived around the most recently selected frame, which we will refer to as the current frame. Note that the local observation input to the policy at each step only captures part of the global state of the inference process, s_t. This choice to limit the observations to only a local window around the current frame is motivated by the desire to allow for π consider long videos of different length and to make fast decisions, since otherwise the intelligent selection would be too costly to pay off.

However, limiting the size of local observation window can lead to sub-optimal decisions if optimal decisions depend on a wider context. To help address this we represent π using a recurrent neural network – namely, an LSTM, which attempts to learn a hidden state that captures the more global context of the inference process. Due to its ability to memorize information from previous decisions made by π, the LSTM has been shown to successfully model problems with non-Markovian state transitions such as ours and have a number of recent empirical successes in sequential decision making [108, 49, 179].

In particular, when the current frame at time t is i, the LSTM-based policy π makes a decision based on:

1) The local information in a video neighborhood $\mathcal{N}_i$ centered around i. This is captured in an observation vector $o_t = [\boldsymbol{z}_{\mathcal{N}_i}, \phi(\mathcal{N}_i), l_t]$, where $\boldsymbol{z}_{\mathcal{N}_i}$ is an indicator vector that indicates whether each frame in $\mathcal{N}_i$ has been previously selected and processed by $\boldsymbol{f}$, $\phi(\mathcal{N}_j)$ is the average of per-class confidences predicted by $\boldsymbol{f}$ in $\mathcal{N}_i$ (yielding one input image per class), and $l_t \in [0, 1]$ is the normalized location of the current frame at time t (e.g. the middle frame has value 0.5). The inclusion of l_t was helpful in encouraging the policy to cover the entire video extent. The input to the LSTM π at step t is o_t.

2) The LSTM's hidden variables $\boldsymbol{h}_{t-1}$, which summarizes the previous observations up to time t.

To summarize, the global state at time t is approximated by the internal state of the LSTM, $\boldsymbol{h}_t$, which depends on the current observation o_t and the previous state $\boldsymbol{h}_{t-1}$. Given h_t the output of $\pi(h_t)$ is the location of the next observation $l_{t+1} \in [0, 1]$. Note that our formulation allows the policy to perform jumps forward and backward in time. Note that π is defined as a probabilistic policy, which is a convenient choice for our policy-gradient learning algorithm defined later. To improve exploration at training time, instead of using l_{t+1}, the next location is sampled from a

Gausssian distribution with a mean equal to l_{t+1} and a fixed variance.

6.4.2 Interpolation Model

The goal of $\boldsymbol{g}$ is to estimate the pixel labels of frames not selected by π. Efficient and reliable interpolation of labels has been demonstrated on videos with smooth motions, i.e., strong correlations between neighboring frames [13, 15, 148]. Intuitively, given the posterior of class labels for frames j in the neighborhood of frame i, and the amount of change observed between frames i and j, our convolution filter $\boldsymbol{g}$ is learned to estimate labels of pixels in i, i.e. to output a posterior distribution over class labels for each pixel in i. The input to $\boldsymbol{g}$ is defined as an ordered set of pixel label predictions for the closest labeled frames j on two sides of frame i along with the additional channels containing pixel-wise frame differences between frames j and i. In our experiments, $\boldsymbol{g}$ is defined as a single layer convolution filter of size 5×5 with $2 \cdot (\#\text{ classes} + 1)$ input channels.

6.4.3 Joint Learning of the Parameters

The goal is to jointly learn the parameters of π and $\boldsymbol{g}$ ($\theta = \{\theta_\pi, \theta_{\boldsymbol{g}}\}$) by minimizing the labeling loss of a sequence of policy actions, taken from the initial state s_0 when no frames are selected until s_T, when the total runtime $C(T) \leq B \leq C(T+1)$.

Recall that $\hat{\boldsymbol{y}}_i^n$ is the estimated output for frame i in video $\boldsymbol{x}^n$. Let $\boldsymbol{u}^{(t)}$ be the indicator vector showing the selected frames in the entire video after running the policy for t steps. We can formally define the estimated output at each time step t as:

$$\hat{\boldsymbol{y}}_i^n(t) = \left[\boldsymbol{u}_{ni}^{(t)} \boldsymbol{f}(\boldsymbol{x}_i^n) + (1 - \boldsymbol{u}_{ni}^{(t)}) \boldsymbol{g}(\boldsymbol{x}^n))\right] \tag{6.4}$$

The main difficulty is that the estimated output $\hat{\boldsymbol{y}}$ for the entire video, is computed through a sequence of decisions made by the policy which results in a non-decomposable, non-differentiable objective function. The decisions that the policy makes at any time depends on a history of decisions that the policy made in previous time steps and influences the decisions available to the policy in the future. This is a long-standing problem in the study of reinforcement learning algorithms. To address this problem, the REINFORCE algorithm [168] and the recurrent policy gradient approach [166] approximate the gradients of the non-decomposable objective function which helps to efficiently learn the policy using stochastic gradient descent.

To follow the general reinforcement learning formulation, let r_t be the immediate reward associated with state s_t. Since $s_t \approx \boldsymbol{h}_t$ we define r_t as :

$$r_t(\boldsymbol{h}_t) = L_\theta(\boldsymbol{y}, \hat{\boldsymbol{y}}(t)) - L_\theta(\boldsymbol{y}, \hat{\boldsymbol{y}}(t-1)), \tag{6.5}$$

where L_θ is the labeling loss for the video defined in eq. (6.2). Intuitively, eq. (6.5) states that the policy earns an immediate reward equal to the decrease in labeling error achieved by selecting a frame (or pay a penalty if the labeling error increases). Let $R_t(H_t)$ be the discounted accumulated reward starting from state s_t and continuing the policy up to final state, s_T:

$$R_t(H_t) = \sum_{t'=t}^{T} \lambda^{t-t'} r_t(\boldsymbol{h}_t), \tag{6.6}$$

where $\lambda \in (0, 1)$ is the discount factor and $H_t = \{\boldsymbol{h}_t, \boldsymbol{h}_{t+1}, ..., \boldsymbol{h}_T\}$ represents a history of LSTM's hidden variables. H_0 can be interpreted as the trajectory of observations for a sample run of the policy from the initial state. For simplification purposes we use H for H_0 and R for R_0 in the rest of this paper. The goal is to find the parameters θ^*, that maximizes $J(\theta)$ redefined as:

$$J(\theta) = E[R(H)] = \int p(H|\theta) R_\theta(H) dH, \tag{6.7}$$

where $p(H|\theta)$ is the probability of observing a sequence of hidden states H, given a policy defined by parameters θ. It is easy to show that minimizing $L(\theta)$ in eq. (6.3) is equivalent to maximizing $J(\theta)$ in eq. (6.7). Let θ_π and $\theta_{\boldsymbol{g}}$ define the gradient of the objective function $J(\theta)$ in eq. (6.7) with respect to the LSTM policy and interpolation networks parameters. Although it is possible to jointly learn the parameters of the LSTM policy and the interpolation networks, given the stochastic nature of the policy, in practice we observed that the iterative approach works better. Alg .4 shows our proposed training procedure:

Computing $\bigtriangledown_{\theta_\pi} J$: The gradient with respect to the policy parameters is given by:

$$\bigtriangledown_{\theta_\pi} J = \int \left[\triangledown_{\theta_\pi} p(H|\theta) R_\theta(H) + p(H|\theta) \triangledown_{\theta_\pi} R_\theta(H)\right] dH \tag{6.8}$$

Note that given the hidden state sequence H, which determines the history of selected frames, the reward function does not depend on the policy parameters, yielding $\bigtriangledown_{\theta_\pi} R_\theta(H) = 0$.

Algorithm 4 Training procedure for our Budget-Aware semantic segmentation model

```
   Input: N Training videos
   Output: Learned parameters $\{\theta_\pi, \theta_{\boldsymbol{g}}\}$.
1: pre-train the interpolation network, $\boldsymbol{g}$ % note: training examples are generated from uniform
   sampling of frames in each video.
2: initialize the policy parameters, $\theta_\pi$
3: for number of iterations do
       % generate trajectories to train $\pi$
4:     $\{H^n\}_{n=1}^N \leftarrow$ aplly $\pi$ for $T$ steps
       % interpolate labels for each video
5:     $\{\hat{\boldsymbol{y}^n}\}_{n=1}^N \leftarrow$ using eq. (6.4)
       % Update policy parameters:
6:     $\theta_\pi \leftarrow - \bigtriangledown_{\theta_\pi} J$
       % generate trajectories to train $\boldsymbol{g}$
7:     $\{H^n\}_{n=1}^N \leftarrow$ aplly $\pi$ for $T$ steps
       % Update interpolation parameters:
8:     $\theta_{\boldsymbol{g}} \leftarrow - \bigtriangledown_{\theta_{\boldsymbol{g}}} J$
9: end for
```

To further simplify (6.8) we need to define $\bigtriangledown_{\theta_\pi} p(H|\theta)$. Note that $p(H|\theta)$ can be factorized as $p(H|\theta) = p(\boldsymbol{h}_0)\prod_{t=1}^T p(\boldsymbol{h}_t|\boldsymbol{h}_{t-1})\pi(l_t|\boldsymbol{h}_{t-1}, o_t)$, where $o_t = [\boldsymbol{z}_{\mathcal{N}_j}^{(t)}, \phi(\boldsymbol{f}_{\mathcal{N}_j}^{(t)}), l_t]$ and the same notation π is used to denote the last softmax layer of the LSTM. From the above we have $\log p(H|\theta) = const + \sum_{t=0}^T \log \pi(l_t|\boldsymbol{h}_{t-1}, o_t)$ which results in the following gradient:

$$\bigtriangledown_{\theta_\pi} \log p(H|\theta) = \sum_{t=0}^T \nabla_\theta \log \pi(l_t|\boldsymbol{h}_{t-1}, o_t)$$

Monte Carlo integration is used to approximate the integration over the probability of observing a sequence of hidden states. Particularly the approximate gradient is computed by running the current policy on N given videos to generate N trajectories which result in the following approximate gradient:

$$\bigtriangledown_{\theta_\pi} J \approx \frac{1}{N}\sum_{n=1}^N \sum_{t=0}^T \big[\bigtriangledown_{\theta_\pi} \log \pi(l_t^n|\boldsymbol{h}_{t-1}^n, o_t^n) R_t(\boldsymbol{h}_t^n)\big]. \tag{6.9}$$

Policy gradient approaches suffer from the high variance of the gradient estimates. Following the common practice [168] we subtract a bias from the expected reward, R. Instead of a constant bias, we set the bias value to be the reward obtained following a random jump policy.

Computing $\bigtriangledown_{\boldsymbol{\theta_g}} J$: Analogous to eq. (6.8), the gradient with respect to the interpolation parameters is defined as $\bigtriangledown_{\theta_g} J = \int \left[\triangledown_{\theta_g} p(H|\theta) R_\theta(H) + p(H|\theta) \triangledown_{\theta_g} R_\theta(H)\right] dH$. Note that since $\boldsymbol{g}$ is applied after frames are selected, the hidden state probability does not depend on the interpolation function, i.e. $\bigtriangledown_{\boldsymbol{\theta_g}} p(H|\theta) = 0$ which results in:

$$\bigtriangledown_{\theta_g} J = \int p(H|\theta) \triangledown_{\theta_g} R_\theta(H) dH. \tag{6.10}$$

Recall that Δ is the average cross-entropy loss for pixels in frame i. It is easy to derive the following gradient:

$$\bigtriangledown_{\theta_g} L_\theta(\boldsymbol{y}, \hat{\boldsymbol{y}}(t)) = \frac{1}{|\boldsymbol{u}^{(t)}|} \sum_{\{i|u_i^{(t)}=0\}} \bigtriangledown_{\boldsymbol{\theta_g}} \Delta(\boldsymbol{y}_i, \boldsymbol{g}(\boldsymbol{x})). \tag{6.11}$$

Intuitively the labeling error of the frames which have not been selected by the policy is considered in computing the gradient with respect to the parameters of $\boldsymbol{g}$. Given a video $\boldsymbol{x}$ and applying (6.5), (6.7), and (6.11), it is easy to derive the following:

$$\bigtriangledown_{\theta_g} R_\theta(H) = \sum_{t=0}^{T} \lambda^t \triangledown_{\theta_g} [L_\theta(\boldsymbol{y}, \hat{\boldsymbol{y}}(t)) - L_\theta(\boldsymbol{y}, \hat{\boldsymbol{y}}(t-1))] \tag{6.12}$$

Using the same Monte Carlo integration technique we derive the following approximate gradient:

$$\bigtriangledown_{\boldsymbol{\theta_g}} J \approx \frac{1}{N} \sum_{n=1}^{N} \sum_{t=0}^{T} \lambda^t \triangledown_{\theta_g} [L_\theta(\boldsymbol{y}^n, \hat{\boldsymbol{y}}^n(t)) - L_\theta(\boldsymbol{y}^n, \hat{\boldsymbol{y}}^n(t-1))] \tag{6.13}$$

6.5 Results

We use the following datasets: 1) CamVid [12], and 2) KITTI [40]. Both datasets are recorded in uncontrolled environment, and present challenges in terms of occlusions, and variations of motions, shapes, and lighting.

Implementation: The LSTM model contains two hidden layers of 1024 hidden units . For training, we generate sequences of continuous frames per each training video where the sequence size is set to be 90 frames in both datasets. Also since two of the video data sets for semantic segmentation do not provide ground truth for all frames (e.g. CamVid provides labels for every 30 frames), the output of $\boldsymbol{f}$ is considered as ground truth for frames without human annotated labels. Note that since the goal is to perform as well as a model that uses $\boldsymbol{f}$ across the entire video, it is reasonable to consider outputs of $\boldsymbol{f}$ as the ground truth. In our evaluation, the budget is defined in terms of the percentage of the maximum required budget needed to apply $\boldsymbol{f}$ for all video frames. Following [37], to improve convergence properties, we start with $\lambda = 0.9$ and gradually update it after each epoch using: $\lambda_{e+1} = 1 - 0.98(1 - \lambda_e)$.

We implement our interpolation and policy modules in tensorflow[2] and use the publicly available code for implementations of $\boldsymbol{f}$. Experiments are performed on an Intel quad core-i7 CPU and 16GB RAM on a single Tesla k80.

Variations of our approach and the baselines: We define two variants for the policy: i) REG: deterministically selects T frames uniformly starting from the first frame, ii) LSTM: learns the proposed model in Sec. 6.4.1. For interpolation we define the following variants: i) OPT: is the baseline approach proposed in [148] which uses dense optical flow to track the points from both forward and backward directions and then propagates the labels from the closest labeled points. More sophisticated label propagation approaches [97, 32, 36] are more expensive and are not suitable as a low-cost interpolation baseline. ii) CNN: learns the proposed interpolation filter in Sec. 6.4.2. The combination of [π = LSTM:$\boldsymbol{g}$ = CNN] is our approach and all other combinations are considered as baselines. We evaluate the validity of our approach on two deep semantic image segmentation models, i) SegNet[9] and ii) BayesianSegNet[73]. We would like to reemphasize that, $\boldsymbol{f}$ is considered as a black-box in our framework. The above two models are only chosen because of the large variation in their accuracies and processing times which allows us to explore the generalizability of our framework in various settings.

6.5.1 Results on CamVid

The CamVid dataset has five videos of road scenes from a moving camera of length up to 6120 frames. Following prior work, we focus on the 11 most common object class labels. Ground-truth labels are available at every 30 frames. We use the standard test-train split as in [9] and similarly resize the frames to 360×280 pixels. The average per frame inference time is 165 ms

[2]https://www.tensorflow.org/

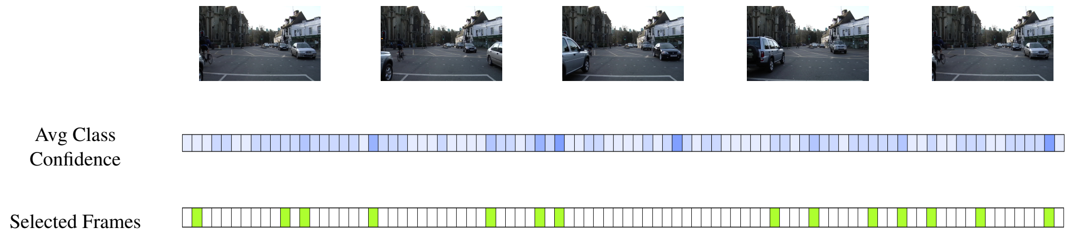


Figure 6.2: An example trajectory of running our learn budget-aware policy on a sample video from CamVid. The top row shows frames 5090 to 05180 from '0016E5' video. The second row shows the color coded average confidence score for each frame where the darker means higher confidence. The third row shows the selected frames by our policy π. The policy is trained for $B = 0.25B_{\text{max}}$ using Bayesian SegNet f =[73].

.

for SegNet and 1450 ms for BayesianSegNet.

Table 6.1 shows the results for three different budgets and $\mathcal{N} = 7$. For $B = 0.1 \cdot B_{\text{max}}$ and $B = 0.25 \cdot B_{\text{max}}$ our [LSTM:CNN] outperforms all other variants in both class average accuracy and mean intersection over union [3]. One interesting observation is the contribution of each module in the accuracy. Based on the result, although both LSTM and CNN provides slight improvement in accuracy when applied independent of each other, they improve the accuracy with a larger margin when learned together. For $B = 0.5 \cdot B_{\text{max}}$, we observe a different pattern. Although our [LSTM:CNN] provides a better result, considering the relative accuracy between [LSTM:OPT] and [REG:OPT] and the same comparison between [REG:CNN] and [REG:OPT] it seems that accuracy boost is mostly due to the interpolation model.

Table 6.2 shows the processing time for policy execution, label interpolation using $\boldsymbol{g}$, and semantic labeling using $\boldsymbol{f}$ for video 'seq16E5' under different budget constraints. We observe that using $\boldsymbol{f}$ =[9], the policy selects almost 30% of the frames when $B = 0.25 \cdot B_{\text{max}}$ and 50% of the frames when $0.5 \cdot B_{\text{max}}$. For $\boldsymbol{f}$ =[73] which takes longer to run, the policy selects only 15% of the frames when $B = 0.25 \cdot B_{\text{max}}$ and 40% of the frames when $0.5 \cdot B_{\text{max}}$. This verifies our main hypothesis that the prior knowledge of the budget changes the behavior of the intelligent system and helps the intelligent agent to learn a particular frame selection pattern that improves the labeling accuracy for that budget.

Fig. 6.3 shows the class-average accuracy for different user-defined budgets. Despite a small accuracy drop, our approach keeps a consistent level of accuracy which shows the effectiveness

[3]Intersection over Union (I/U) for one class = $\frac{tp}{tp+fp+fn}$

B	Method	Road	Building	Sky	Tree	Side-Walk	Car	Column-Pole	Fence	Pedestrian	Bicycle	Sign	Class Avg	Mean I/U
							f = SegNet [9]							
$B_{\max}$	All Frames	88.0	87.3	92.3	80.0	29.5	97.6	57.2	49.4	27.8	84.8	30.7	65.9	50.2
$0.1 \cdot B_{\max}$	[REG:OPT]	56.4	46.4	53.7	42.7	10.3	60.3	29.3	18.5	9.5	36.8	10.7	34.1	26.7
	[REG:CNN]	57.2	47.0	51.5	43.8	11.2	60.9	28.4	19.3	9.8	37.9	11.1	34.4	26.9
	[LSTM:OPT]	60.8	50.3	54.9	51.2	16.8	70.1	32.7	25.6	15.7	44.2	14.7	39.7	30.2
	[LSTM:CNN]	63.4	58.8	61.9	54.9	17.3	73.9	38.1	31.3	19.5	51.7	20.8	44.7	33.8
$0.25 \cdot B_{\max}$	[REG:OPT]	70.4	69.1	73.2	63.8	23.6	81.9	45.4	39.1	20.9	65.9	23.8	52.5	40.2
	[REG:CNN]	70.8	68.9	75.4	64.2	24.8	78.6	45.8	39.4	22.1	66.8	24.5	52.8	40.5
	[LSTM:OPT]	83.4	70.4	83.7	72.3	25.4	88.1	52.3	45.2	24.6	78.6	26.8	59.2	45.4
	[LSTM:CNN]	81.1	80.3	82.9	73.7	27.6	89.8	54.2	45.9	25.8	78.0	28.4	60.7	46.2
$0.5 \cdot B_{\max}$	[REG:OPT]	83.2	82.6	86.3	75.4	27.5	93.2	53.7	46.8	25.9	80.1	29.1	62.2	47.0
	[REG:CNN]	81.6	82.1	86.0	75.7	27.4	90.8	53.1	46.9	25.8	81.4	27.6	61.7	46.9
	[LSTM:OPT]	81.5	82.0	86.7	75.8	27.1	91.4	52.9	46.4	25.2	80.6	27.5	61.6	46.7
	[LSTM:CNN]	84.1	83.7	87.4	76.2	27.9	93.4	54.3	46.3	26.3	80.5	29.3	62.7	47.7
							f = Bayesian SegNet [73]							
$B_{\max}$	All Frames	80.4	85.5	90.1	86.4	67.9	93.8	73.8	64.5	50.8	91.7	54.6	76.3	63.1
$0.1 \cdot B_{\max}$	[REG:OPT]	51.6	49.3	59.6	41.9	44.7	46.9	34.4	35.8	24.5	45.2	20.7	41.3	34.8
	[REG:CNN]	52.7	52.8	57.8	44.7	46.2	50.3	36.8	36.9	25.2	46.7	21.6	42.9	36.1
	[LSTM:OPT]	59.2	57.8	63.2	54.9	49.1	57.6	42.4	40.1	31.3	56.7	30.5	49.3	40.8
	[LSTM:CNN]	60.3	60.1	64.8	56.7	50.3	60.1	46.8	42.3	33.7	59.4	31.6	51.5	42.3
$0.25 \cdot B_{\max}$	[REG:OPT]	60.8	65.8	70.1	66.3	51.6	70.8	57.1	49.7	38.3	70.9	41.5	58.4	47.9
	[REG:CNN]	62.5	65.2	71.5	68.4	52.3	71.2	60.3	52.2	39.9	70.4	41.9	59.6	49.3
	[LSTM:OPT]	76.3	81.1	83.8	81.4	63.1	87.5	68.9	60.9	47.3	85.1	51.3	71.5	58.7
	[LSTM:CNN]	76.0	80.9	85.7	82.8	64.6	88.1	70.4	61.2	48.8	84.6	52.5	72.3	59.4
$0.5 \cdot B_{\max}$	[REG:OPT]	75.7	80.4	83.8	82.1	64.3	89.5	69.5	60.2	48.2	86.3	51.3	71.9	59.0
	[REG:CNN]	75.8	80.9	86.4	82.5	64.8	89.8	69.5	61.1	48.5	87.5	51.8	72.6	59.6
	[LSTM:OPT]	75.2	80.5	85.6	82.4	63.9	89.0	70.2	60.1	47.9	85.8	51.1	72.0	58.8
	[LSTM:CNN]	77.1	81.9	86.2	81.7	65.1	88.7	69.3	61.8	49.1	88.2	52.8	72.9	59.8

Table 6.1: Comparison with different variations of our budget-aware inference on CamVid. The budget is defined as a fraction of the maximum budget required to run the original methods, [9, 73] for each frame, denoted as $B_{\max}$

Method	Time for π	Time for $\boldsymbol{g}$	Time for $\boldsymbol{f}$	Class-Avg
$\boldsymbol{f}$ = SegNet [9]				
REG+CNN($0.25 \cdot B_{\text{max}}$)	0	130.4	251.3	53.5
REG+CNN($0.5 \cdot B_{\text{max}}$)	0	90.6	567.2	61.7
LSTM+CNN($0.25 \cdot B_{\text{max}}$)	23.1	125.8	256.7	61.7
LSTM+CNN($0.5 \cdot B_{\text{max}}$)	5.2	92.1	441.6	62.8
$\boldsymbol{f}$= Bayesian SegNet [73]				
REG+CNN($0.25 \cdot B_{\text{max}}$)	0	184.5	2928.4	62.3
REG+CNN($0.5 \cdot B_{\text{max}}$)	0	97.3	4170.8	72.6
LSTM+CNN($0.25 \cdot B_{\text{max}}$)	20.4	196.8	2934.7	72.3
LSTM+CNN($0.5 \cdot B_{\text{max}}$)	8.8	113.9	4181.3	73.3

Table 6.2: Comparison of the total processing time for different variations of our framework on CamVid for $B = 0.25B_{\text{max}}$.

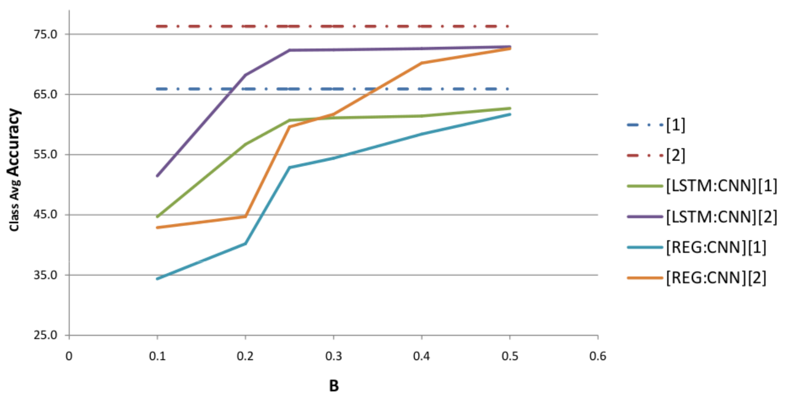


Figure 6.3: The class-average accuracy for different budget constraints using [1] =[73], [2] =[9].

of the learned frame selection and interpolation when $\frac{1}{4}B_{\text{max}} < B < \frac{1}{2}B_{\text{max}}$.

Fig. 6.2 shows the percentage of the frames selected in different temporal regions of video ‘0016E5’ in CamVid. For better visualization, only 90 frames (frames 5090 to 05180) are presented. the video is divided into equal segments of 10 frames. To better reflect the behavior of the policy with respect to its observations, we provide a color coded average class confident for each temporal segment.

Qualitative results on the CamVid dataset are shown in Fig .6.4. The columns represent four consecutive frames of a sample video.

Results on KITTI: The KITTI dataset consists of videos from road scenes where eight

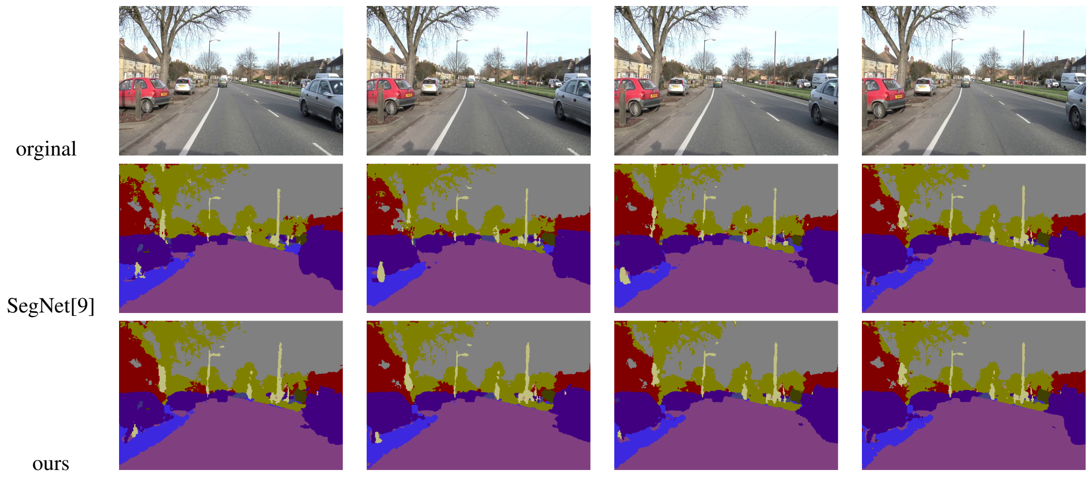


Figure 6.4: Sample results from video '0006R0' on CamVid. The top row shows the input and the middle row shows the result of applying SegNet on individual frames. The bottom row shows the outputs of our LSTM:CNN approach. While only the frame in the fourth column is selected in our approach, qualitatively the results look very similar for other frames.

class labels, (Building, Tree, Sign, Road, Fence, Pole, Sidewalk), are annotated in a subset of frames. Since the number of ground-truth annotations is much less than the number total of frames in videos. Instead of comparing with ground truth, for evaluating KITTI, we compare our approach with a method which applies $\boldsymbol{f}$ on all video frames. Considering the fact that we are ultimately upper-bounded by the accuracy of the full run of $\boldsymbol{f}$ on the entire video this is a reasonable evaluation. Note that that during training we also use the very same outputs from applying $\boldsymbol{f}$ as the ground truth for training the policy and the interpolation. As in [73], we resize the images size to 360×280 and re-train only the fourth layer of the SegNet. Table 6.3 shows the per-class and class-average accuracy compared to results obtain from $\boldsymbol{f}$. For $B = 0.25 \cdot B_{\max}$ our budget-aware inference achieves 88.4% accuracy and for $B = 0.5 \cdot B_{\max}$ we achieve 90% accuracy. For a 4-fold speed up we have an accuracy reduction of only 11.5%.

6.6 Summary

We have addressed the problem of budgeted semantic video segmentation, where pixels of a video must be labeled within a time budget. We have specified a budget-aware inference for this problem that intelligently selects a subset of frames to run a deep CNN for semantic segmenta-

Budget	Method	Building	Tree	Sign	Road	Fence	Pole	Sidewalk	Class Avg
$0.25 \cdot B_{\max}$	[REG:OPT]	89.7	81.2	74.2	83.3	61.5	72.2	85.3	78.2
	[REG:CNN]	90.5	82.3	77.4	86.9	67.3	74.7	86.8	80.8
	[LSTM:OPT]	91.2	86.8	80.2	90.2	71.2	78.4	89.2	83.9
	[LSTM:CNN]	91.9	92.1	89.5	93.5	78.3	83.2	90.2	88.4
$0.5 \cdot B_{\max}$	[REG:OPT]	92.2	91.2	91.8	95.8	88.9	87.5	93.9	91.6
	[REG:CNN]	94.4	90.6	91.1	94.1	89.7	87.1	91.8	91.3
	[LSTM:OPT]	93.7	89.3	90.8	93.7	89.7	86.6	90.3	90.6
	[LSTM:CNN]	93.0	94.1	92.6	96.1	90.2	89.1	94.0	92.7

Table 6.3: Comparison of different variation of our budget-aware inference on KITTI. The budget is defined as a fraction of the maximum budget required to run the original method, [9] for each frame, denoted as $B_{\max}$.

tion. Since CNN computation often dominates the cost of inference, our framework can provide substantial time savings in a principled manner. For selecting the subset of frames, we have formulated a visual-attention policy within the MDP framework, and used an LSTM as the policy model. We have also specified a new segmentation propagation function to label the unselected frames as a one-layer CNN. Our experiments show that our approach significantly improves on the accuracies of several strong baselines. The results also demonstrate that we can optimally adapt our method, from operating with no time bound to varying time budgets, such that it yields satisfactory performance for one-forth of the maximum budget, while maintaining an accuracy as close as possible to its performance for no time bound.

Chapter 7: Concluding Remarks and Future Work

In this thesis, we addressed the problems of human action classification, summarization and semantic labeling in videos captured in real-world settings. We made a number of contributions.

First, we improved the state of the art performance on multi-view action classification using a multi-task learning framework in which each task represents a single viewpoint. We specified a group regularization for our Latent Multitask Learning (LMTL). LMTL identifies groupings of correlated viewpoints and jointly regularizes the models in the same group to identify the most informative feature subspace for each group. Finally, the parameters of the entire model are learned using a coordinate descent algorithm. We verified that in the case of a limited number of training instances in a subset of viewpoints our model is able to perform comparable to a model which accesses the entire data.

Second, we showed that the accuracy of the state of the art deep learning action classification is improved using a limited number of additional training data from a different modality, particularly 3D human skeleton sequences. Our novel deep multimodal learning framework which consists of a video model and a 3D model learns to classify videos by exploiting representation of the encoded 3D skeletons sequences. Specifically, our 3D model is an encoder LSTM (eLSTM) which is used to encode human skeleton sequences and is trained in a standard encoder/decoder framework. Our video model is an LSTM network grounded on top of frame-level DCNN features with an additional video representation layer which is used to introduce class-specific and class independent similarity constraints. We proposed a new hybrid gradient descent learning for training regularized LSTM networks. We verified that the accuracy improvement is not because of the amount of the additional training data but its modality.

Third, we illustrated that our unsupervised video summarization approach, built on top of variational recurrent auto-encoders and generative adversarial networks (GAN) performs equally as good as the state of the art supervised approaches. The main hypothesis is that the learned representation of the summary video and the original video should be similar. Two subnetworks of our model, 'summarizer' and the 'discriminator' are trained in an adversarial manner where the summarizer aims to summarize the video such that the discriminator is fooled and the discriminator aims to recognize the summary videos from original videos. Particularly we use the

GAN's discriminator to learn a discrete similarity measure. Through extensive evaluation, we verified that even without the supervision of keyframes we are able to achieve comparable accuracy to the state of the art supervised video summarization approaches. We also showed that in the case of additional supervision we outperform the state of the art.

Fourth, we defined the new problem of budget-aware semantic video segmentation, where the pixels of a video must be semantically labeled under a time budget. We verified that a learned intelligent policy helps in reaching a tradeoff between accuracy and efficiency. Since the tradeoff between accuracy and efficiency is not well-defined, we believe an intelligent system should be informed about the users' expectations regarding the acceptable tradeoff. We formulated the budgeted inference in a reinforcement learning framework and proposed two different models. Our first model is grounded on the traditional CRF architecture. Since descriptor computation often dominates the cost of CRF inference, our framework provides substantial time savings by formulating the inference policy for selecting among descriptors to run for each supervoxel in a video. Classification-Based Policy Iteration (CAPI) is used to learn the policy.

Unlike the CRF-Based scene labeling the state of the art uses Convolutional Neural Network (CNN) for pixel labeling at the frame level. Since applying CNN is the most expensive computation, we reduced the processing cost by limiting the number of times CNN is applied in a video. We leveraged the fact that videos typically show smooth motions, and hence pixel labels of a frame can be efficiently and accurately interpolated from neighboring segmentations. A policy is learned to select a subset of frames to apply CNN and an interpolation network is learned to propagate the label posterior to the remaining frames. Particularly we used a Long Short-Term Memory (LSTM) to implement the policy, where the LSTM's architecture allows it to memorize the partial observations in time. We verified that our budget-aware semantic segmentation achieves 4X speedup for a 10% accuracy drop.

Bibliography

[1] Carnegie-Mellon Mocap. urlhttp://mocap.cs.cmu.edu. Accessed: 2003.

[2] Open video project. url http://www.open-video.org/.

[3] YouTube Statistics. urlhttps://www.youtube.com/yt/press/statistics.html. Accessed: 2016.

[4] Mohamed Amer, Dan Xie, Mingtian Zhao, Sinisa Todorovic, and Song-Chun Zhu. Cost-sensitive top-down/bottom-up inference for multiscale activity recognition. In *ECCV*, 2012.

[5] Mohamed R Amer and Sinisa Todorovic. Sum-product networks for modeling activities with stochastic structure. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1314–1321. IEEE, 2012.

[6] Mohamed R. Amer, Sinisa Todorovic, Alan Fern, and Song-Chun Zhu. Monte carlo tree search for scheduling activity recognition. In *ICCV*, 2013.

[7] Aya Aner and John R. Kender. *Video Summaries through Mosaic-Based Shot and Scene Clustering*, pages 388–402. Springer Berlin Heidelberg, Berlin, Heidelberg, 2002.

[8] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Convex multi-task feature learning. *Machine Learning*, 73:243–272, 2008.

[9] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *arXiv preprint arXiv:1511.00561*, 2015.

[10] Chen Bo. Deep learning of invariant spatio-temporal features from video. 2010.

[11] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(11):1222–1239, 2001.

[12] Gabriel J Brostow, Jamie Shotton, Julien Fauqueur, and Roberto Cipolla. Segmentation and recognition using structure from motion point clouds. In *ECCV*. 2008.

[13] Ignas Budvytis, Vijay Badrinarayanan, and Roberto Cipolla. Label propagation in complex video sequences using semi-supervised learning. In *BMVC*, 2010.

[14] Rich Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997.

[15] Albert YC Chen and Jason J Corso. Propagating multi-class pixel labels throughout video frames. In *WNYIPW*, 2010.

[16] Albert YC Chen and Jason J Corso. Temporally consistent multi-class video-object segmentation with the video graph-shifts algorithm. In *WACV*, 2011.

[17] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. *arXiv preprint arXiv:1412.7062*, 2014.

[18] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv preprint arXiv:1606.00915*, 2016.

[19] Ronan Collobert, Samy Bengio, and Johnny Marithoz. Torch: A modular machine learning software library, 2002.

[20] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[21] Navneet Dalal, Bill Triggs, and Cordelia Schmid. Human detection using oriented histograms of flow and appearance. In *Computer Vision–ECCV 2006*, pages 428–441. Springer, 2006.

[22] Sandra Eliza Fontes De Avila, Ana Paula Brandão Lopes, Antonio da Luz, and Arnaldo de Albuquerque Araújo. Vsumm: A mechanism designed to produce static video summaries and a novel evaluation method. *Pattern Recognition Letters*, 32(1):56–68, 2011.

[23] Christos Dimitrakakis and Michail G Lagoudakis. Rollout sampling approximate policy iteration. *Machine Learning*, 72(3):157–171, 2008.

[24] Piotr Dollár, Vincent Rabaud, Garrison Cottrell, and Serge Belongie. Behavior recognition via sparse spatio-temporal features. In *VS-PETS workshop*, pages 65–72. IEEE, 2005.

[25] Jeff Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. *arXiv preprint arXiv:1411.4389*, 2014.

[26] Yong Du, Wei Wang, and Liang Wang. Hierarchical recurrent neural network for skeleton based action recognition. In *CVPR*, pages 1110–1118, 2015.

[27] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 12:2121–2159, 2011.

[28] Mohamed Elhoseiny, Burhan Saleh, and Ahmed Elgammal. Write a classifier: Zero-shot learning using purely textual descriptions. In *ICCV*, pages 2584–2591. IEEE, 2013.

[29] Otto Fabius and Joost R van Amersfoort. Variational recurrent auto-encoders. *arXiv preprint arXiv:1412.6581*, 2014.

[30] Ali Farhadi and Mostafa Kamali Tabrizi. Learning to recognize activities from the wrong view point. In *ECCV*, 2008.

[31] Ali Farhadi, Mostafa Kamali Tabrizi, Ian Endres, and David A. Forsyth. A latent model of discriminative aspect. In *ICCV*, 2009.

[32] Julien Fauqueur, Gabriel Brostow, and Roberto Cipolla. Assisted video object labeling by joint tracking of regions and keypoints. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–7. IEEE, 2007.

[33] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *PAMI*, 32(9):1627–45, 2010.

[34] Alan Fern, Sung Wook Yoon, and Robert Givan. Approximate policy iteration with a policy language bias. In *NIPS*, 2003.

[35] Alan Fern, Sung Wook Yoon, and Robert Givan. Approximate policy iteration with a policy language bias: Solving relational markov decision processes. *JAIR*, 25:75–118, 2006.

[36] Philipp Fischer, Alexey Dosovitskiy, Eddy Ilg, Philip Häusser, Caner Hazırbaş, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. *arXiv preprint arXiv:1504.06852*, 2015.

[37] Vincent François-Lavet, Raphael Fonteneau, and Damien Ernst. How to discount deep reinforcement learning: Towards new dynamic strategies. *arXiv preprint arXiv:1512.02011*, 2015.

[38] Andrea Frome, Greg S Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Tomas Mikolov, et al. Devise: A deep visual-semantic embedding model. In *NIPS*, pages 2121–2129, 2013.

[39] Marco Furini, Filippo Geraci, Manuela Montangero, and Marco Pellegrini. Stimo: Still and moving video storyboard for the web scenario. *Multimedia Tools and Applications*, 46(1):47–69, 2010.

[40] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, pages 3354–3361. IEEE, 2012.

[41] Mathieu Gerard, Bart De Schutter, and Michel Verhaegen. A hybrid steepest descent method for constrained convex optimization. *Automatica*, 45(2):525–531, 2009.

[42] Arnab Ghosh, Viveka Kulharia, Amitabha Mukerjee, Vinay Namboodiri, and Mohit Bansal. Contextual rnn-gans for abstract reasoning diagram generation. *arXiv preprint arXiv:1609.09444*, 2016.

[43] Nikolaos Gkalelis, Hansung Kim, Adrian Hilton, Nikos Nikolaidis, and Ioannis Pitas. The i3DPost multi-miew and 3D human action/interaction database. *CVMP*, 2009.

[44] Dan B Goldman, Brian Curless, Steven M. Seitz, and David Salesin. Schematic storyboarding for video visualization and editing. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 25(3), July 2006. [to appear].

[45] Boqing Gong, Wei-Lun Chao, Kristen Grauman, and Fei Sha. Diverse sequential subset selection for supervised video summarization. In *Advances in Neural Information Processing Systems*, pages 2069–2077, 2014.

[46] Dian Gong and Gerard Medioni. Dynamic manifold warping for view invariant action recognition. *ICCV*, 2011.

[47] Dian Gong and Gerard Medioni. Dynamic manifold warping for view invariant action recognition. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 571–578. IEEE, 2011.

[48] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

[49] Karol Gregor, Ivo Danihelka, Alex Graves, and Daan Wierstra. Draw: A recurrent neural network for image generation. *arXiv preprint arXiv:1502.04623*, 2015.

[50] Alexander Grubb, Daniel Munoz, J Andrew Bagnell, and Martial Hebert. Speedmachines: Anytime structured prediction. *arXiv preprint arXiv:1312.0579*, 2013.

[51] Matthias Grundmann, Vivek Kwatra, Mei Han, and Irfan Essa. Efficient hierarchical graph-based video segmentation. In *CVPR*, 2010.

[52] Michael Gygli, Helmut Grabner, Hayko Riemenschneider, and Luc Van Gool. Creating summaries from user videos. In *ECCV*, 2014.

[53] Michael Gygli, Helmut Grabner, Hayko Riemenschneider, and Luc Van Gool. Creating summaries from user videos. In *European conference on computer vision*, pages 505–520. Springer, 2014.

[54] Michael Gygli, Helmut Grabner, and Luc Van Gool. Video summarization by learning submodular mixtures of objectives. In *CVPR*, pages 3090–3098, 2015.

[55] Hossein Hajimirsadeghi and Greg Mori. Learning ensembles of potential functions for structured prediction with latent variables. In *ICCV*, 2015.

[56] Tavi Halperin, Yair Poleg, Chetan Arora, and Shmuel Peleg. Egosampling: Wide view hyperlapse from single and multiple egocentric videos. *CoRR*, abs/1604.07741, 2016.

[57] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Hypercolumns for object segmentation and fine-grained localization. In *CVPR*, pages 447–456, 2015.

[58] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[59] Cisco Visual Networking Index. Forecast and methodology, 2015-2020 white paper. *Retrieved 01 Jun*, 2016.

[60] Alexandros Iosifidis, Nikos Nikolaidis, and Ioannis Pitas. Movement recognition exploiting multi-view information. In *MMSP*, 2010.

[61] Aastha Jain, Shuanak Chatterjee, and René Vidal. Coarse-to-fine semantic video segmentation using supervoxel trees. In *ICCV*, 2013.

[62] Ashesh Jain, Hema Swetha Koppula, Bharad Raghavan, and Ashutosh Saxena. Know before you do: Anticipating maneuvers via learning temporal driving models. *CoRR*, abs/1504.02789, 2015.

[63] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3d convolutional neural networks for human action recognition. *PAMI*, 35(1):221–231, 2013.

[64] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[65] Neel Joshi, Wolf Kienzle, Mike Toelle, Matt Uyttendaele, and Michael F. Cohen. Real-time hyperlapse creation via optimal frame selection. *ACM Trans. Graph.*, 34(4):63:1–63:9, July 2015.

[66] Imran N Junejo, Emilie Dexter, Ivan Laptev, and Patrick Pérez. View-independent action recognition from temporal self-similarities. *PAMI*, 33(1):172–85, 2011.

[67] Andrew Kae, Benjamin Marlin, and Erik Learned-Miller. The shape-time random field for semantic video labeling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 272–279, 2014.

[68] Zhuoliang Kang and Kristen Grauman. Learning with whom to share in multi-task feature learning. *ICML*, 2011.

[69] J.H. Kappes, M. Speth, G. Reinelt, and C. Schnorr. Towards efficient and exact map-inference for large scale discrete computer vision problems via combinatorial optimization. In *CVPR*, 2013.

[70] S. Karayev, M. Fritz, and T. Darrell. Anytime recognition of objects and scenes. In *CVPR*, 2014.

[71] Andrej Karpathy, George Toderici, Sachin Shetty, Tommy Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, pages 1725–1732. IEEE, 2014.

[72] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014.

[73] Alex Kendall, Vijay Badrinarayanan, , and Roberto Cipolla. Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding. *arXiv preprint arXiv:1511.02680*, 2015.

[74] Aditya Khosla, Raffay Hamid, Chih-Jen Lin, and Neel Sundaresan. Large-scale video summarization using web-image priors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2698–2705, 2013.

[75] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[76] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *ICLR*, 2014.

[77] Diederik P Kingma and Max Welling. Stochastic gradient vb and the variational auto-encoder. *Talk Slides*, 2014.

[78] Ryan Kiros, Ruslan Salakhutdinov, and Richard S Zemel. Unifying visual-semantic embeddings with multimodal neural language models. *arXiv preprint arXiv:1411.2539*, 2014.

[79] Johannes Kopf, Michael F. Cohen, and Richard Szeliski. First-person hyper-lapse videos. *ACM Trans. Graph.*, 33(4):78:1–78:10, July 2014.

[80] Philipp Krähenbühl and Vladlen Koltun. Efficient inference in fully connected CRFs with gaussian edge potentials. In *NIPS*, 2012.

[81] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: a large video database for human motion recognition. In *ICCV*, 2011.

[82] Abhijit Kundu, Yin Li, Frank Dellaert, Fuxin Li, and James M Rehg. Joint semantic segmentation and 3d reconstruction from monocular video. In *ECCV*. 2014.

[83] Michail Lagoudakis and Ronald Parr. Reinforcement learning as classification: Leveraging modern classifiers. In *ICML*, 2003.

[84] Tian Lan, Leonid Sigal, and Greg Mori. Social roles in hierarchical models for human activity recognition. In *Computer Vision and Pattern Recognition (CVPR)*, 2012.

[85] Tian Lan, Yuke Zhu, Amir Roshan Zamir, and Silvio Savarese. Action recognition by hierarchical mid-level action elements. In *ICCV*, 2015.

[86] Ivan Laptev. On space-time interest points. *IJCV*, 64(2-3):107–123, 2005.

[87] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. *arXiv preprint arXiv:1512.09300*, 2015.

[88] Alessandro Lazaric, Mohammad Ghavamzadeh, and Rémi Munos. Analysis of a classification-based policy iteration algorithm. In *ICML*, 2010.

[89] Yong Jae Lee, Joydeep Ghosh, and Kristen Grauman. Discovering important people and objects for egocentric video summarization. In *2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, June 16-21, 2012*, pages 1346–1353. IEEE Computer Society, 2012.

[90] Peng Lei and Sinisa Todorovic. Recurrent temporal deep field for semantic video labeling. In *European Conference on Computer Vision*, pages 302–317. Springer, 2016.

[91] Ruonan Li and Todd Zickler. Discriminative virtual views for cross-view action recognition. *CVPR*, 2012.

[92] Yingbo Li and Bernard Merialdo. Multi-video summarization based on video-mmr. In *11th International Workshop on Image Analysis for Multimedia Interactive Services WIAMIS 10*, pages 1–4. IEEE, 2010.

[93] Ming Liang, Xiaolin Hu, and Bo Zhang. Convolutional neural networks with intra-layer recurrent connections for scene labeling. In *NIPS*, pages 937–945, 2015.

[94] Buyu Liu and Xuming He. Multiclass semantic video segmentation with object-level active inference. *Journal of Machine Learning Research*, 4:1107–1149, 2003.

[95] Buyu Liu and Xuming He. Learning dynamic hierarchical models for anytime scene labeling. *ECCV*, 2016.

[96] Buyu Liu, Xuming He, and Stephen Gould. Multi-class semantic video segmentation with exemplar-based object reasoning. In *WACV*, 2015.

[97] Ce Liu, Jenny Yuen, and Antonio Torralba. Nonparametric scene parsing: Label transfer via dense scene alignment. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1972–1979. IEEE, 2009.

[98] Jingen Liu, Jiebo Luo, and Mubarak Shah. Recognizing realistic actions from videos in the wild. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1996–2003. IEEE, 2009.

[99] Jingen Liu, Mubarak Shah, Benjamin Kuipers, and Silvio Savarese. Cross-view action recognition via view knowledge transfer. *CVPR*, 2011.

[100] Xiao Liu, Dacheng Tao, Mingli Song, Ying Ruan, Chun Chen, and Jiajun Bu. Weakly supervised multiclass video segmentation. In *CVPR*, 2014.

[101] Nicolas Loeff and Ali Farhadi. Scene discovery by matrix factorization. In *ECCV*, 2008.

[102] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[103] Jiajia Luo, Wei Wang, and Hairong Qi. Group sparsity and geometry constrained dictionary learning for action recognition from depth maps. In *ICCV*, pages 1809–1816. IEEE, 2013.

[104] Fengjun Lv and Ramakant Nevatia. Single view human action recognition using key pose matching and viterbi path searching. In *CVPR*, 2007.

[105] Junhua Mao, Wei Xu, Yi Yang, Jiang Wang, and Alan Yuille. Deep captioning with multimodal recurrent neural networks (m-rnn). *arXiv preprint arXiv:1412.6632*, 2014.

[106] Michael Mathieu, Camille Couprie, and Yann LeCun. Deep multi-scale video prediction beyond mean square error. *arXiv preprint arXiv:1511.05440*, 2015.

[107] Ondrej Miksik, Daniel Munoz, J Andrew Bagnell, and Martial Hebert. Efficient temporal consistency for streaming video scene analysis. In *ICRA*, 2013.

[108] Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. Recurrent models of visual attention. In *NIPS*, pages 2204–2212, 2014.

[109] Padmavathi Mundur, Yong Rao, and Yelena Yesha. Keyframe-based video summarization using delaunay clustering. *International Journal on Digital Libraries*, 6(2):219–232, 2006.

[110] Meinard Mller, Tido Rder, Michael Clausen, Bernhard Eberhardt, Bjrn Krger, and Andreas Weber. Documentation mocap database hdm05, 2007.

[111] Natalia Neverova, Christian Wolf, Graham W. Taylor, and Florian Nebout. Moddrop: adaptive multi-modal gesture recognition. *CoRR*, abs/1501.00102, 2015.

[112] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond short snippets: Deep networks for video classification. *arXiv preprint arXiv:1503.08909*, 2015.

[113] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. *CoRR*, abs/1505.04366, 2015.

[114] Mohammad Norouzi, Tomas Mikolov, Samy Bengio, Yoram Singer, Jonathon Shlens, Andrea Frome, Greg S Corrado, and Jeffrey Dean. Zero-shot learning by convex combination of semantic embeddings. *arXiv preprint arXiv:1312.5650*, 2013.

[115] Vasu Parameswaran and Rama Chellappa. View invariance for human action recognition. *IJCV*, 66(1):83–101, 2006.

[116] Pedro HO Pinheiro and Ronan Collobert. Recurrent convolutional neural networks for scene parsing. *arXiv preprint arXiv:1306.2795*, 2013.

[117] John C Platt and Alan H Barr. Constrained differential optimization for neural networks. 1988.

[118] Yair Poleg, Tavi Halperin, Chetan Arora, and Shmuel Peleg. Egosampling: Fast-forward and stereo for egocentric videos. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[119] Danila Potapov, Matthijs Douze, Zaid Harchaoui, and Cordelia Schmid. Category-specific video summarization. In *ECCV 2014 - European Conference on Computer Vision*, Zurich, Switzerland, September 2014. Springer.

[120] Y. Pritch, A. Rav-Acha, A. Gutman, and S. Peleg. Webcam synopsis: Peeking around the world. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8, Oct 2007.

[121] Y. Pritch, A. Rav-Acha, and S. Peleg. Nonchronological video synopsis and indexing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(11):1971–1984, Nov 2008.

[122] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1994.

[123] Ariadna Quattoni, Michael Collins, and Trevor Darrell. Transfer learning for image classification with sparse prototype representations. In *CVPR*, 2008.

[124] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[125] MarcAurelio Ranzato, Arthur Szlam, Joan Bruna, Michael Mathieu, Ronan Collobert, and Sumit Chopra. Video (language) modeling: a baseline for generative models of natural videos. *arXiv preprint arXiv:1412.6604*, 2014.

[126] Cen Rao, Alper Yilmaz, and Mubarak Shah. View-invariant representation and recognition of actions. *IJCV*, 50(2):203–226, 2002.

[127] Alex Rav-Acha, Yael Pritch, and Shmuel Peleg. Making a long video short: Dynamic video synopsis. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2006), 17-22 June 2006, New York, NY, USA*, pages 435–441. IEEE Computer Society, 2006.

[128] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. *arXiv preprint arXiv:1605.05396*, 2016.

[129] Gemma Roig, Xavier Boix, Roderick De Nijs, Sebastian Ramos, Koljia Kuhnlenz, and Luc Van Gool. Active MAP inference in CRFs for efficient semantic segmentation. In *ICCV*, 2013.

[130] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, pages 1–42, April 2015.

[131] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *arXiv preprint arXiv:1606.03498*, 2016.

[132] Evan Shelhamer, Kate Rakelly, Judy Hoffman, and Trevor Darrell. Clockwork convnets for video semantic segmentation. *ECCV*, 2016.

[133] Yuping Shen and Hassan Foroosh. View-invariant action recognition from point triplets. *PAMI*, 31(10):1898–905, 2009.

[134] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, pages 568–576, 2014.

[135] Richard Socher, Milind Ganjoo, Christopher D Manning, and Andrew Ng. Zero-shot learning through cross-modal transfer. In *NIPS*, pages 935–943, 2013.

[136] Yale Song, Jordi Vallmitjana, Amanda Stent, and Alejandro Jaimes. Tvsum: Summarizing web videos using titles. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5179–5187, 2015.

[137] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

[138] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhutdinov. Unsupervised learning of video representations using lstms. *arXiv preprint arXiv:1502.04681*, 2015.

[139] Min Sun, Ali Farhadi, Ben Taskar, and Steve Seitz. *Salient Montages from Unconstrained Videos*, pages 472–488. Springer International Publishing, Cham, 2014.

[140] Eran Swears, Anthony Hoogs, Qiang Ji, and Kim Boyer. Complex activity recognition using granger constrained dbn (gcdbn) in sports and surveillance video. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 788–795. IEEE, 2014.

[141] Tanveer Fathima Syeda-Mahmood, M. Alex O. Vasilescu, and Saratendu Sethi. Recognizing action events from multiple viewpoints. In *IEEE Workshop on Detection and Recognition of Events in Video*, pages 64–, 2001.

[142] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *arXiv preprint arXiv:1409.4842*, 2014.

[143] Brian Taylor, Alper Ayvaci, Avinash Ravichandran, and Stefano Soatto. Semantic video segmentation from occlusion relations within a convex optimization framework. In *Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 195–208. Springer, 2013.

[144] Graham W Taylor, Rob Fergus, Yann LeCun, and Christoph Bregler. Convolutional learning of spatio-temporal features. In *Computer Vision–ECCV 2010*, pages 140–153. Springer, 2010.

[145] Gerald Tesauro and Gregory R Galperin. On-line policy improvement using monte-carlo search. In *NIPS*, 1996.

[146] Antonio Torralba, Kevin P. Murphy, and William T. Freeman. Sharing visual features for multiclass and multiview object detection. *PAMI*, 29(5):854–869, 2007.

[147] Du Tran, Lubomir D. Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. C3D: generic features for video analysis. *CoRR*, abs/1412.0767, 2014.

[148] Sudheendra Vijayanarasimhan and Kristen Grauman. Active frame selection for label propagation in videos. In *European Conference on Computer Vision*, pages 496–509. Springer, 2012.

[149] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating videos with scene dynamics. *arXiv preprint arXiv:1609.02612*, 2016.

[150] Chunyu Wang, Yizhou Wang, and Alan L Yuille. An approach to pose-based action recognition. In *CVPR*, pages 915–922. IEEE, 2013.

[151] Heng Wang, Alexander Kläser, Cordelia Schmid, and Cheng-Lin Liu. Action Recognition by Dense Trajectories. In *IEEE Conference on Computer Vision & Pattern Recognition*, pages 3169–3176, Colorado Springs, United States, June 2011.

[152] Heng Wang, Alexander Kläser, Cordelia Schmid, and Cheng-Lin Liu. Action recognition by dense trajectories. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3169–3176. IEEE, 2011.

[153] Heng Wang, Alexander Klaser, Cordelia Schmid, and Cheng-Lin Liu. Action recognition by dense trajectories. In *CVPR*, 2011.

[154] Heng Wang, Alexander Kläser, Cordelia Schmid, and Cheng-Lin Liu. Dense trajectories and motion boundary descriptors for action recognition. *IJCV*, 103(1):60–79, 2013.

[155] Heng Wang and Cordelia Schmid. Action recognition with improved trajectories. In *ICCV*, pages 3551–3558. IEEE, 2013.

[156] Heng Wang, Muhammad Muneeb Ullah, Alexander Klaser, Ivan Laptev, and Cordelia Schmid. Evaluation of local spatio-temporal features for action recognition. In *BMVC*, pages 124–1. BMVA Press, 2009.

[157] Jiang Wang, Zicheng Liu, Ying Wu, and Junsong Yuan. Mining actionlet ensemble for action recognition with depth cameras. In *CVPR*, pages 1290–1297. IEEE, 2012.

[158] Limin Wang, Yu Qiao, and Xiaoou Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. *arXiv preprint arXiv:1505.04868*, 2015.

[159] Daniel Weinland, Edmond Boyer, and Remi Ronfard. Action recognition from arbitrary views using 3D Exemplars. *ICCV*, 2007.

[160] Daniel Weinland, Mustafa Özuysal, and Pascal Fua. Making action recognition robust to occlusions and viewpoint changes. *ECCV*, 2010.

[161] Daniel Weinland, Remi Ronfard, and Edmond Boyer. Free viewpoint action recognition using motion history volumes. *CVIU*, 104(2):249–257, 2006.

[162] David Weiss, Benjamin Sapp, and Ben Taskar. Dynamic structured model selection. In *ICCV*, 2013.

[163] David J Weiss and Ben Taskar. Learning adaptive value of information for structured prediction. In *NIPS*. 2013.

[164] Hong wen Kang, Yasuyuki Matsushita, Xiaoou Tang, and Xue quan Chen. Space-time video montage. In *In CVPR06*, pages 1331–1338, 2006.

[165] Jason Weston, Frédéric Ratle, Hossein Mobahi, and Ronan Collobert. Deep learning via semi-supervised embedding. In *Neural Networks: Tricks of the Trade*, pages 639–655. Springer, 2012.

[166] Daan Wierstra, Alexander Förster, Jan Peters, and Jürgen Schmidhuber. Recurrent policy gradients. *Logic Journal of IGPL*, 18(5):620–634, 2010.

[167] Geert Willems, Tinne Tuytelaars, and Luc Van Gool. An efficient dense and scale-invariant spatio-temporal interest point detector. In *ECCV*, pages 650–663. Springer, 2008.

[168] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[169] Christian Wojek, Stefan Roth, Konrad Schindler, and Bernt Schiele. Monocular 3d scene modeling and inference: Understanding multi-object traffic scenes. In *ECCV*, pages 467–481. Springer, 2010.

[170] Christian Wojek and Bernt Schiele. A dynamic conditional random field model for joint labeling of object and scene classes. In *ECCV*. 2008.

[171] Di Wu and Ling Shao. Leveraging hierarchical parametric networks for skeletal joints based action segmentation and recognition. In *CVPR*, pages 724–731. IEEE, 2014.

[172] T. Wu and S.-C. Zhu. Learning near-optimal cost-sensitive decision policy for object detection. In *ICCV*, 2013.

[173] Xinxiao Wu and Yunde Jia. View-invariant action recognition using latent kernelized structural svm. In *ECCV*, 2012.

[174] Ying Wu. Mining actionlet ensemble for action recognition with depth cameras. In *CVPR*, 2012.

[175] Lu Xia, Chia-Chih Chen, and JK Aggarwal. View invariant human action recognition using histograms of 3d joints. In *CVPR Workshop*, pages 20–27. IEEE, 2012.

[176] Chenliang Xu, Spencer Whitt, and Jason J Corso. Flattening supervoxel hierarchies by the uniform entropy slice. In *ICCV*, 2013.

[177] Pingkun Yan, Saad M Khan, and Mubarak Shah. Learning 4D action feature models for arbitrary view action recognition. *ICCV*, 2008.

[178] Huan Yang, Baoyuan Wang, Stephen Lin, David Wipf, Minyi Guo, and Baining Guo. Unsupervised extraction of video highlights via robust recurrent auto-encoders. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4633–4641, 2015.

[179] Serena Yeung, Olga Russakovsky, Greg Mori, and Li Fei-Fei. End-to-end learning of action detection from frame glimpses in videos. *arXiv preprint arXiv:1511.06984*, 2015.

[180] A. Yilmaz and M. Shah. Actions sketch : A novel action representation. *CVPR*, 2005.

[181] Chun-Nam John Yu and Thorsten Joachims. Learning structural SVMs with latent variables. *ICML*, 2009.

[182] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

[183] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014.

[184] Shengxin Zha, Florian Luisier, Walter Andrews, Nitish Srivastava, and Ruslan Salakhutdinov. Exploiting image-trained cnn architectures for unconstrained video classification. *arXiv preprint arXiv:1503.04144*, 2015.

[185] Ke Zhang, Wei-Lun Chao, Fei Sha, and Kristen Grauman. Summary transfer: Exemplar-based subset selection for video summarization. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[186] Ke Zhang, Wei-Lun Chao, Fei Sha, and Kristen Grauman. Video summarization with long short-term memory. *arXiv preprint arXiv:1605.08110*, 2016.

[187] Tianzhu Zhang, Bernard Ghanem, Si Liu, and Narendra Ahuja. Robust visual tracking via structured multi-task sparse learning. *IJCV*, 101(2):367–383, 2012.

[188] Yimeng Zhang and Tsuhan Chen. Efficient inference for fully-connected CRFs with stationarity. In *CVPR*, 2012.

[189] Bin Zhao and Eric P Xing. Quasi real-time summarization for consumer videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2513–2520, 2014.

[190] Junbo Zhao, Michael Mathieu, and Yann LeCun. Energy-based generative adversarial network. *arXiv preprint arXiv:1609.03126*, 2016.

[191] Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip Torr. Conditional random fields as recurrent neural networks. *arXiv preprint arXiv:1502.03240*, 2015.

APPENDICES

Appendix A: Redundancy

This appendix is inoperable.